Interpretable Fine-Grained Visual Categorization

Pei Guo

Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Physical Sciences and Mathematics Commons

BYU ScholarsArchive Citation


This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.
ABSTRACT

Interpretable Fine-Grained Visual Categorization

Pei Guo
Department of Computer Science, BYU
Doctor of Philosophy

Not all categories are created equal in object recognition. Fine-grained visual categorization (FGVC) is a branch of visual object recognition that aims to distinguish subordinate categories within a basic-level category. Examples include classifying an image of a bird into specific species like “Western Gull” or “California Gull”. Such subordinate categories exhibit characteristics like small inter-category variation and large intra-class variation, making distinguishing them extremely difficult. To address such challenges, an algorithm should be able to focus on object parts and be invariant to object pose. Like many other computer vision tasks, FGVC has witnessed phenomenal advancement following the resurgence of deep neural networks. However, the proposed deep models are usually treated as black boxes. Network interpretation and understanding aims to unveil the features learned by neural networks and explain the reason behind network decisions. It is not only a necessary component for building trust between humans and algorithms, but also an essential step towards continuous improvement in this field.

This dissertation is a collection of papers that contribute to FGVC and neural network interpretation and understanding. Our first contribution is an algorithm named Pose and Appearance Integration for Recognizing Subcategories (PAIRS) which performs pose estimation and generates a unified object representation as the concatenation of pose-aligned region features. As the second contribution, we propose the task of semantic network interpretation. For filter interpretation, we represent the concepts a filter detects using an attribute probability density function. We propose the task of semantic attribution using textual summarization that generates an explanatory sentence consisting of the most important visual attributes for decision-making, as found by a general Bayesian inference algorithm. Pooling has been a key component in convolutional neural networks and is of special interest in FGVC. Our third contribution is an empirical and experimental study towards a thorough yet intuitive understanding and extensive benchmark of popular pooling approaches. Our fourth contribution is a novel LMPNet for weakly-supervised keypoint discovery. A novel leaky max pooling layer is proposed to explicitly encourages sparse feature maps to be learned. A learnable clustering layer is proposed to group the keypoint proposals into final keypoint predictions. 2020 marks the 10th year since the beginning of fine-grained visual categorization. It is of great importance to summarize the representative works in this domain. Our last contribution is a comprehensive survey of FGVC containing nearly 200 relevant papers that cover 7 common themes.

Keywords: fine-grained visual categorization, pose-aligned representation, global pooling, network interpretation, network understanding, weakly-supervised keypoint discovery
I would like to thank my advisor, Ryan Farrell, for his relentless support and help for both my academic research and personal life. He is always there when I have questions to ask and ideas to discuss about. My Ph.D. journey has been challenging but rewarding. Dr. Farrell is the one who makes this journey as smooth and enjoyable as possible. He has always been encouraging, positive and nice to everyone around him.

I am grateful for the time and efforts my dissertation committee members, David Wingate, Bryan Morse, and Parris Egbert, have put into reviewing my dissertation. I thank Conner Anderson for his help in discussing interesting ideas, providing valuable feedback and carrying out some important experiments.

I thank my family in China for their love and support during the past six years. Last but not the least, I thank my beautiful wife Yajing Zhao, for her love and companion in this special period of my life. I can’t imagine what my life will be without her. Yajing, I love you and will always belong to you.
# Table of Contents

Abstract

Acknowledgements

List of Figures

List of Tables

1 Introduction

References

2 Aligned to the Object, not to the Image: A Unified Pose-aligned Representation for Fine-grained Recognition

2.1 Introduction

2.2 Background and Related Work

2.3 **PAIRS** - Pose and Appearance Integration

2.3.1 Pose Estimation Network

2.3.2 Patch Generation

2.3.3 Patch Feature Extraction

2.3.4 Classification Network

2.4 Experimental Evaluation

2.4.1 Keypoint Prediction Performance

2.4.2 Patch Classification Network
2.4.3 Classification Network .................................................. 20
2.4.4 Additional Study .......................................................... 23
2.4.5 Results Visualization ..................................................... 24

2.5 Conclusion ................................................................. 25

References ............................................................................. 26

3 Semantic Network Interpretation ........................................... 33

3.1 Introduction ...................................................................... 34
3.2 Related Work .................................................................... 37
3.3 Bayesian Inference Framework ......................................... 39
  3.3.1 Filter-Attribute Distribution ......................................... 39
  3.3.2 Textual Summarization ................................................ 42
3.4 Experiments ..................................................................... 43
  3.4.1 Network Debugging ...................................................... 45
  3.4.2 Human Study ............................................................ 45
  3.4.3 Network Understanding ............................................... 47
3.5 Conclusion ...................................................................... 50

References ............................................................................. 51

4 On Global Feature Pooling for Fine-grained Visual Categorization 56

4.1 Introduction ...................................................................... 57
4.2 Pooling Schemes Overview .............................................. 62
4.3 A Comparison of Max and Average Pooling ...................... 64
4.4 Experiments ..................................................................... 67
  4.4.1 Global Pooling Evaluation ........................................... 68
  4.4.2 Perceptual Sparsity for K-max Pooling ......................... 71
  4.4.3 Heterogeneous Pooling Integration .............................. 72
References

5 LMPNet for Weakly-supervised Keypoint Discovery 79

5.1 Introduction ........................................... 80
5.2 Related Work ......................................... 84
5.3 LMPnet Architecture ................................. 86
  5.3.1 Leaky Max Pooling ................................. 87
  5.3.2 Filter Consistency ................................. 89
  5.3.3 Filter Diversity ................................. 91
  5.3.4 Learnable Clustering ............................. 92
5.4 Experiment ........................................... 93
  5.4.1 Leaky Max Pooling Analysis .................... 93
  5.4.2 Keypoint Prediction Evaluation .................. 95
5.5 Conclusion ........................................... 96

References 98

6 Fine-grained Visual Categorization (FGVC): A Comprehensive 10-year Survey 103

6.1 Introduction ........................................... 104
6.2 Background and Related Areas ..................... 107
  6.2.1 Categorization Spectrum ....................... 108
  6.2.2 Perceptual Expertise .......................... 109
  6.2.3 Object Morphology and Recognition .......... 109
  6.2.4 Instance Recognition and Biometrics .......... 111
  6.2.5 Deep Learning .................................. 111
6.3 Themes and Strategies ............................. 112
<table>
<thead>
<tr>
<th>Section Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.1 Segmentation</td>
<td>113</td>
</tr>
<tr>
<td>6.3.2 Human-In-the-Loop</td>
<td>117</td>
</tr>
<tr>
<td>6.3.3 Features and Pooling</td>
<td>120</td>
</tr>
<tr>
<td>6.3.4 Parts and Part-Normalization</td>
<td>124</td>
</tr>
<tr>
<td>6.3.5 Attention</td>
<td>134</td>
</tr>
<tr>
<td>6.3.6 Hierarchy/Taxonomy</td>
<td>138</td>
</tr>
<tr>
<td>6.3.7 Misc</td>
<td>141</td>
</tr>
<tr>
<td>6.4 Datasets and Performance Evaluation</td>
<td>147</td>
</tr>
<tr>
<td>6.4.1 Dataset Overview</td>
<td>150</td>
</tr>
<tr>
<td>6.4.2 Workshop Competitions</td>
<td>154</td>
</tr>
<tr>
<td>6.4.3 Category-wise Accuracy</td>
<td>157</td>
</tr>
<tr>
<td>6.5 FGVC Workshops and FGVC.org Website</td>
<td>157</td>
</tr>
<tr>
<td>6.6 Conclusion and Future Directions</td>
<td>159</td>
</tr>
<tr>
<td>References</td>
<td>160</td>
</tr>
<tr>
<td>7 Conclusion</td>
<td>187</td>
</tr>
<tr>
<td>7.1 Future Directions</td>
<td>188</td>
</tr>
<tr>
<td>References</td>
<td>190</td>
</tr>
</tbody>
</table>
## List of Figures

1.1 Challenges in FGVC ............................................................. 2

2.1 Motivation for pose-aligned regions ....................................... 9

2.2 Filter visualization ............................................................ 10

2.3 The proposed framework ..................................................... 14

2.4 Pose-aligned Patch Generation ............................................ 16

2.5 Keypoint Detection Results .................................................. 19

2.6 Patch accuracy comparison .................................................. 23

2.7 Result visualization ........................................................... 24

3.1 Semantic network interpretation .......................................... 34

3.2 Examples of filter visualization ............................................ 36

3.3 Filter-level semantic interpretation ....................................... 39

3.4 Algorithm overview .......................................................... 41

3.5 Textual summarization examples .......................................... 42

3.6 Network debugging examples .............................................. 44

3.7 Quantitative analysis ........................................................ 46

3.8 Filter selectivity study ........................................................ 48

4.1 Comparison between average and max pooling ....................... 57

4.2 Pooling study ................................................................. 67

4.3 Generalization gap analysis ............................................... 70

5.1 Final convolutional layer filter visualization ......................... 81
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Pose estimation results</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Performance on CUB</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>Performance on NABirds</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>Human study results</td>
<td>46</td>
</tr>
<tr>
<td>4.1</td>
<td>Nine popular pooling methods</td>
<td>60</td>
</tr>
<tr>
<td>4.2</td>
<td>Pooling benchmark</td>
<td>66</td>
</tr>
<tr>
<td>4.3</td>
<td>Heterogeneous pooling integration</td>
<td>71</td>
</tr>
<tr>
<td>5.1</td>
<td>Classification Accuracy for different $\epsilon$</td>
<td>94</td>
</tr>
<tr>
<td>5.2</td>
<td>Greedy PCK results</td>
<td>96</td>
</tr>
<tr>
<td>6.1</td>
<td>Dataset overview</td>
<td>148</td>
</tr>
<tr>
<td>6.2</td>
<td>Results on the CUB-200-2010 Dataset</td>
<td>149</td>
</tr>
<tr>
<td>6.3</td>
<td>Results on Flowers-102 Dataset</td>
<td>150</td>
</tr>
<tr>
<td>6.4</td>
<td>Results on Stanford Cars Dataset</td>
<td>154</td>
</tr>
<tr>
<td>6.5</td>
<td>Results on Aircraft Dataset</td>
<td>155</td>
</tr>
<tr>
<td>6.6</td>
<td>Results on the CUB-200-2011 Dataset</td>
<td>156</td>
</tr>
<tr>
<td>6.7</td>
<td>Results on Stanford Dogs Dataset</td>
<td>157</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Object recognition is a fundamental task in computer vision whose goal is to classify an object into its correct category. In recent years, object recognition has witnessed dramatic performance improvement ignited by the resurgence of deep neural networks and the availability of large-scale datasets. However, not all categories are created equal in object recognition: some categories are inherently more difficult than others to distinguish, both for humans and for computer algorithms. To demonstrate this, Deng, et al. [3] provide a good visualization of the confusion matrix produced by the SPM+SVM algorithm on the ImageNet dataset. The majority of errors concentrate in fine-grained categories of birds, dogs, and cars, etc. Moreover, human performance on the ImageNet dataset is also impacted by a lack of the knowledge needed to differentiate fine-grained categories [5]. Fine-grained visual categorization (FGVC) is a branch of object recognition that aims to distinguish subordinate categories (e.g. Tesla Model 3 vs. Tesla Model Y and Common Tern vs. Arctic Tern) within a basic-level category (e.g. cars and birds). FGVC has become an important research direction in modern computer vision, not only because of the unique challenges it presents, but also for its diverse real-world applications. Examples include popular mobile apps that help people identify species of birds, plants, etc.

Several inherent characteristics make fine-grained recognition more challenging than general image recognition. On one hand, the inter-class variation can be small (see Figure 1.1). A domain expert is often needed to discriminate between very similar species that, to an average observer, appear to be identical. The only differences between two species of gulls are
the colors of the eyes and the feet. Deng, et al. [3] reported a strong correlation between the structure of the semantic hierarchy (based on WordNet [4]) and the visual confusion among categories.

On the other hand, intra-class variation can be high due to dramatic changes in external factors such as background and lighting or intrinsic factors such as pose and appearance. Birds provide a prime example of the intrinsic factors. A single species can exhibit significant variation in appearance depending on gender, age/maturity and even the season of the year. Males and females often differ in their appearance; males tend to be more colorful and flashy in appearance to draw attention during courtship, while females generally have more drab and nondescript plumage to help conceal them while nesting. Juvenile birds often display a distinct plumage relative to their adult counterparts; some birds such as gulls have unique plumages each year until they reach sexual maturity. Further, many species have different plumages during their breeding season and varying degrees of transitional plumage can be observed before and after.

To overcome inter-category variation, an algorithm should focus on object parts. To eliminate intra-category variation, the features of object parts should be invariant to object
pose. This leads to our first contribution: the PAIRS (Pose and Appearance Integration for Recognizing Subcategories) algorithm. According to Bengio et al. [1], a good feature representation should disentangle object pose and appearance. PAIRS achieves this via a unified object representation built upon pose-aligned regions. These regions are characterized by rectangular patches defined by and aligned relative to two keypoint anchors. The final object representation is an aggregation of the features extracted from all pose-aligned regions. This representation comprises a pose-invariant and over-complete basis of features at multiple scales.

The convolutional neural network (CNN) [6] is the key model behind most of the successful algorithms in FGVC. However, CNNs are often viewed as black boxes. It is therefore vital to interpret what features have been learned by a network in order to understand how to learn better features. We propose to associate each filter in the final convolutional layer with the attributes that represent its activation patterns. The filter-attribute relationship is modelled as a conditional multinomial probability distribution, represented by the filter attribute probability density function (aPDF). A direct application of the resulting textual explanation algorithm is network debugging, generating descriptive error messages when the network’s prediction is wrong. The filter aPDF provides a means for us to quantitatively understand how concepts are encoded across filters. More importantly, we are able to explore the correlation between a model’s performance with the distributed level of its representation. We examine two aspects of distributed representation: filter selectivity and concept sparseness [2].

Global feature pooling has played an important role in modern networks but its influence on the underlying features has remained largely unexplored. There is an interesting contrast in how max- and average-pooling compared before and after the deep learning breakthrough. A deeper understanding and a thorough benchmark of the available global pooling schemes are necessary. We found that max pooling produces much sparser feature maps and helps the network learn part-level features. Average pooling, on the other hand,
encourages object-level features to be learned. We evaluated nine representative global pooling schemes for fine-grained recognition. K-max (k=2) pooling outperforms all other global pooling schemes and is actually better than unmodified second-order pooling models. Max pooling performs better than average pooling across datasets, models, and input resolutions. Max pooling also generalizes better (exhibits a smaller train-test gap) than average pooling. Classification performance exhibits a monotonic increase as generalized pooling is weighted from favoring average- towards favoring max-pooling. We additionally highlight the importance of post-global batch normalization — which is absent from most, if not all, popular state-of-the-art models — in attaining faster convergence and in consistently improving a model’s performance.

Annotating images with keypoints can be a time-consuming task. A promising direction is to leverage weaker category labels for keypoint discovery. Global max pooling is shown to be effective at detecting part-level features. This characteristic can be exploited further in order to detect keypoint-level features. We devise a general form of pooling as matrix-vector multiplication for which max and average pooling can be seen as special cases. This gives rise to a new form of pooling that further encourages feature map sparsity. This form, called leaky max pooling, is defined as a weight matrix that adaptively adjusts to the input distribution. It allows only a one pixel output like max pooling, but also suppresses other pixels with negative weights. The proposed pooling scheme is shown qualitatively and quantitatively to explicitly encourage sparse feature maps. To group the keypoint proposals into final keypoint predictions, we propose a learnable clustering layer that can be embedded seamlessly into a classification network and can be learned in an end-to-end fashion. The layer weights are updated to reflect the distance between the inputs and outputs, such that a keypoint prediction is connected to a keypoint proposal only when they represent the same keypoint. An erasing step is performed to avoid duplication among the keypoints produced.

As a relatively new research direction that began emerging around the year 2010, a thorough survey paper for fine-grained visual categorization is absent from the literature.
Our survey paper contains 200+ related papers and summarizes the algorithms into six common schemes: foreground segmentation, human-in-the-loop, part-based model, features and pooling, attention, and transfer learning. One can have a better idea how different schemes evolve over time and see a clear transition of research interests. We also identify shortcomings in the literature and propose impactful directions for future works in the field of FGVC.
References


Chapter 2

Aligned to the Object, not to the Image: A Unified Pose-aligned Representation for Fine-grained Recognition

Abstract

Dramatic appearance variation due to pose constitutes a great challenge in fine-grained recognition, one which recent methods using attention mechanisms or second-order statistics fail to adequately address. Modern CNNs typically lack an explicit understanding of object pose and are instead confused by entangled pose and appearance. In this paper, we propose a unified object representation built from pose-aligned regions of varied spatial sizes. Rather than representing an object by regions aligned to image axes, the proposed representation characterizes appearance relative to the object’s pose using pose-aligned patches whose features are robust to variations in pose, scale and viewing angle. We propose an algorithm that performs pose estimation and forms the unified object representation as the concatenation of pose-aligned region features, which is then fed into a classification network. The proposed algorithm attains state-of-the-art results on two fine-grained datasets, notably 89.2% on the widely-used CUB-200 [46] dataset and 87.9% on the much larger NABirds [45] dataset. Our success relative to competing methods shows the critical importance of disentangling pose and appearance for continued progress in fine-grained recognition.
2.1 Introduction

What makes fine-grained visual categorization (FGVC), commonly referred to as fine-grained recognition, different from general visual categorization? One important distinction lies in the difficulty of the datasets. General-purpose visual categorization often involves the classification of everyday objects, such as chairs, bicycles and dogs, which are easy for humans to identify. Fine-grained recognition, on the other hand, consists of more detailed classifications such as identifying the species of a bird. This is extremely difficult for non-expert humans as it requires familiarity with domain knowledge and hundreds of hours of training. Computer algorithms for fine-grained recognition have the potential to be far more accurate than most humans and can thus benefit millions of people by providing services like species recognition through mobile applications [2, 7, 28].

An intrinsic challenge of fine-grained recognition is small inter-category variance coupled with large intra-category variance. Discriminative features for two visually similar categories often lie in a few key locations; while the appearances of two objects from the same category may dramatically different due simply to pose variation. The entangling of appearance and pose presents a great challenge and motivates the need for stable appearance features that are invariant to variations in pose, scale and rotation.

It’s almost instinct for humans to identify and visually compare key locations across objects in different poses, establishing correspondences. Convolutional neural networks, however, struggle on this task because the convolutional mechanisms are purely appearance-based and lack an understanding of geometry or pose. Their built-in pooling mechanisms can tolerate a certain amount of scale and rotation variation [4–6, 36, 47], but exactly how much is still largely an open question [39]. We show this in Figure 2.2 via the visualization of some final convolutional layer responses. We show the top-activated images together with the feature map as a masked region. It’s evident that this convolutional filter is attuned to red beaks. However, due to its lack of pose-awareness, this filter also fires strongly at visually
Figure 2.1: Motivation for Pose-Aligned Regions. Two terns of the same species (images on left), but in different poses, have dramatically different appearances while two different species of woodpecker (images on right) appear nearly identical except for the barring pattern on the outer tail (and the shape of the beak). The large intra-class variance and small inter-class variance of the full images make the feature space distance inaccurately reflect the true class relationships. Such observations motivate the use of pose-aligned regions that disentangle intrinsic part appearance from variations in object pose, leading to a feature space that facilitates correctly classifying the species or fine-grained category.

similar parts such as red crowns, red throats, red eyes, etc. This causes confusion for the classifier because of the noisy entangled pose-appearance representation.

In the feature embedding space, dramatic pose variation can make images of the same category farther separated and images of different but visually-similar categories appear closer together as shown in Figure 2.1. It is therefore vital that pose-aligned regions, which explicitly factor out pose variation, should be the building block of the disentangled image representation.

Recent efforts in fine-grained recognition have largely focused on two directions. One is algorithms related to second-order statistics[8, 16, 24, 33]. Representative works include Bilinear Pooling[33], its memory efficient variants [16, 24], and those that extend to higher-order statistics [8]. The idea is to project the features into a higher-order space where they
Figure 2.2: **Filter Visualization.** We visualize the 25 images that maximally activate an example filter. Different semantic parts (e.g., red beak, red eye, *et al.*) with similar appearance can all activate this filter, causing confusion for the classifier when such semantic parts would be discriminative. This problem can be solved by disentangling pose and appearance.

can be linearly separated. Such methods have both sound theoretical support and work well in practice. However, they look at the image globally, and thus having little hope of finding subtle highly-localized differences. Also, they lack interpretability and insights for further improvement.

The second direction is attention-based methods [15, 29, 35, 41, 48, 57] that use subnetworks to propose possible discriminative regions to attend to. However, the regions proposed by these networks are often weakly-supervised by some heuristic loss function, lacking proof that they really attend to the right location. Both of these directions suffer from a lack of pose awareness and moreover the entanglement of pose and appearance features limits their performance. Furthermore, training data is often scarce in the long-tailed distributions seen in many fine-grained domains; in such cases, both techniques suffer as the limited training imagery does not adequately span the space of pose and viewing angle for each category, hindering their ability to recognize any species in any pose.

Based on the above observations, we propose to disentangle pose and appearance via a unified object representation built upon pose-aligned regions. Those regions are characterized by rectangular patches defined relative to two keypoints anchors. The final object representation is an aggregation of the features from all pose-aligned regions. This
representation comprises a pose-invariant and over-complete basis of features at multiple scales. We contrast the pose-aligned regions with weakly-supervised regions that are generated in a purely data-driven fashion and with “axis-aligned” rectangular bounding boxes centered around a keypoint or landmark. The features from these types of regions are subject to natural variations in pose, scale and viewing angles. We experimentally demonstrate that axis-aligned regions are less-capable of classifying fine-grained datasets compared to pose-aligned regions (see Figure 3.8b).

To automate the process of applying the unified object representation to fine-grained recognition, we propose an algorithm that first performs pose-estimation via keypoint detection, enabling the generation of pose-aligned region features. The local features from these aligned regions, regions of varying size/scale relative to the object, are concatenated to comprise the unified representation for the object and are then fed into a classification network to produce a final classification prediction. We call the proposed algorithm PAIRS: Pose and Appearance Integration for Recognizing Subcategories. It achieves state-of-the-art results on two key fine-grained datasets: CUB-200-2011 [46] and NABirds [45]. Keypoint annotations are used only during training. In consideration of the annotation cost, keypoint annotations may actually be less expensive and time-consuming than collecting additional training samples because keypoints can be annotated by human non-experts whereas fine-grained image category annotations require the consensus of multiple domain experts.

2.2 Background and Related Work

Fine-grained visual categorization (FGVC) lies between generic category-level object recognition like VOC [12], ImageNet [40], COCO [32], etc. and instance-level classification like face recognition or other visual biometrics. The challenges inherent to FGVC are many. Differences between similar species are often subtle and highly-localized and thus difficult even for (non-expert) humans to identify. Dramatic pose changes introduce great intra-class
variance. Generalization also becomes an issue as the network struggles to find truly useful and discriminative features.

FGVC has drawn broad interest within the computer vision community. Early work includes [10, 13, 34, 50–53], two of which explicitly tackle the challenging interplay of pose and appearance. Birdlets [13], a volumetric poselet representation, was proposed to account for the pose and appearance variation. Zhang, et al. [53] further proposed pose-normalized descriptors based on computationally-efficient deformable part models. While they seek to address pose and appearance, their hand-engineered features result in limited success.


Our work is perhaps most closely related to POOF [1] which also uses keypoint pair patches. Unlike POOF, we employ a fully-convolutional network for keypoint detection. And where POOF computes 5000 patch features per image, whereas we’re only computing 35-70.

Other methods focus on object alignment. Unlike previous methods which relied on detectors for part localization, Gavves et al. [17, 18] propose to localize distinctive details by roughly aligning the objects using just the overall shape. Spatial Transformer Networks [23] introduced a differentiable affine transformation learning layer to transform and align the object or part of interest.

Another direction in fine-grained recognition is feature correlation and kernel mapping. Bilinear Pooling [33] computes a second order-polynomial kernel mapping on CNN features.
Several extensions [8, 16, 24] followed this simple paradigm. Compact Bilinear Pooling [16] proposes a compact representation to approximate the polynomial kernel, reducing memory usage. Low-rank Bilinear Pooling [24] represents the covariance features as a matrix and applies a low-rank bilinear classifier. Kernel Pooling [8] proposes a general pooling framework that captures the higher-order interactions of features in the form of kernels. This line of work achieves relatively good results with only weak supervision. These approaches, however, attend to the image globally, lacking a mechanism for part-level discovery. This constrains their potential for further improvement.

Inspired by human attention mechanisms, many attempts have been made to guide the attention of CNN models toward informative object parts. Works along this direction include [15, 29, 35, 41, 48, 57]. Zheng et al. [58] proposes a multi-attention convolutional neural network (MA-CNN), where part generation and feature learning can reinforce each other. Lam et al. [29] leverages long short-term memory (LSTM) networks to unify new patch candidate generation and informative part evaluation. This work establishes the current state-of-the-art performance on CUB-200-2011 dataset, achieving an accuracy of 87.5% with part annotations (excluding works like [9, 27] that utilize outside training data). The key difference in our PAIRS representation is that it integrates pose and appearance information and explicitly achieves multi-scale attention over semantic object parts at the same time.

2.3 **PAIRS - Pose and Appearance Integration**

We illustrate our algorithm pipeline in Figure 2.3. We first apply a simple yet effective fully-convolutional neural network for keypoint-based pose estimation. We follow the prevailing modular design paradigm by stacking convolutional blocks that have similar topology. We show that our pose estimation network achieves superior results on the CUB-200 dataset, both qualitatively and quantitatively. Second, given detected keypoint locations, a rectangular region is aligned to each keypoint pair and cropped from the original image. The region is then similarity-transformed into a uniform-sized patch (see Figure 2.4), such that both
Figure 2.3: **Overview of the Proposed Framework for Fine-grained Recognition.**

We first apply a pose-estimation network to the image for keypoint detection. Pose-aligned regions are then extracted from the image using the predicted keypoint locations. We then extract features from the individual regions using region-specific networks. The concatenated features collectively form a unified multi-scale representation that is invariant to pose, scale and rotation change. This representation is then fed into a classification network for the final fine-grained classification.

Keypoints are at fixed positions across different images. As the representation is normalized to the keypoint locations, the patches are well-aligned, independent of the pose or the camera’s angle. Third, we train a separate CNN model as the feature extractor for each pose-aligned patch representation. Last, we explore different classification architectures for the unified representation based on the assumption that parts differ in their respective contributions for different images and classes. We find surprisingly that the Multi-Layer Perception (MLP), while perhaps the most simple method, achieves the best final classification accuracy.

### 2.3.1 Pose Estimation Network

Pose estimation networks usually follow one of two paradigms for prediction. The first is to directly regress discrete keypoint coordinates, e.g. \((x_i, y_i)\). Representative approaches include [44]. The alternate approach [43] instead uses a two-dimensional probability distribution heat map to represent the keypoint location. We call this resulting multi-channel probability distribution matrix a **pose tensor**.

In this paper, we adopt the second strategy, proposing a fully convolutional network to produce the desired pose tensor. Specifically, we take a pretrained classification network
and remove the final classifier layer(s), retaining what can be seen as an encoder network that encodes strong visual semantics. We follow the prevailing modular design to stack repeated building blocks to the end of the network. This building block consists of one upsampling layer, one convolutional layer, one batch normalization layer and one ReLU layer. The parameter-free bilinear interpolation layer is used for upsampling. The convolutional layer uses 1x1 kernel and reduces the input channel size by half. Last, a final convolutional layer and upsampling layer are added to produce the pose tensor. There are many modifications one can make to enhance this basic model, including using larger 3x3 kernels, adding more convolutional layers to the building block, adding residue connection to each block, stacking more building blocks, and using a learnable transposed convolutional layer for upsampling. We find these structures provide only limited improvement but introduce more parameters, and we therefore adopt this simpler architecture.

2.3.2 Patch Generation

Historically, part-based representations would model parts either as rectangular regions [14, 54] or keypoints. Keypoints are convenient for pose-estimation. However, the square or rectangular patches, each centered on a given keypoint and extracted to characterize the part’s appearance, are far from optimal in the presence of rotation or more general pose variation. We instead, propose to use keypoint pairs as anchor points in extracting pose-aligned patches.

Given two keypoints $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$, we define the vectors $r_{ij} = p_j - p_i$, and $\hat{r}_{ij} = r_{ij}/||r_{ij}||$. We also define the vector $\hat{t}_{ij} = \hat{z} \times \hat{r}_{ij}$, a unit vector perpendicular to $r_{ij}$, and the distances $d = ||r_{ij}||_2$ and $h = d/2$ for convenience. We seek to extract a region around $p_i$ and $p_j$ that is aligned with $r_{ij}$ and has dimensions $2d \times d$. The four corners of this
Figure 2.4: **Pose-aligned Patch Generation.** For each pair of keypoints, we fit a rectangular region whose corners are calculated as in (b). Objects in different poses and/or from different viewpoints can be compared directly by proposed keypoint pair patches shown in (a). Details are described in Section 2.3.2.

A similarity-transform is computed from these corners to extract the pose-normalized patch. Patches generated in this way contain stable pose-aligned features – features near these keypoints appear at roughly the same location in a given patch across different images, independent of the object’s pose or the camera viewing angle. Details are shown in Figure 2.4.

$$
\begin{bmatrix}
(p_i - h\hat{r}_{ij}) + h\hat{t}_{ij} & (p_j + h\hat{r}_{ij}) + h\hat{t}_{ij} \\
(p_i - h\hat{r}_{ij}) - h\hat{t}_{ij} & (p_j + h\hat{r}_{ij}) - h\hat{t}_{ij}
\end{bmatrix}
$$

(2.1)

A separate patch classification network is trained for each posed-aligned $A | B$ patch as a feature extractor ($A$ and $B$ are keypoints). The softmax outputs from those networks are concatenated as the representation for the input image. Alternately, the final convolutional layer output after pooling can also be used and the result is comparable. We find that leveraging symmetry can help reduce the overall number of classifiers by nearly 50%, described in Section 2.4.2.
The proposed patch representation can be seen as a spatial pyramid that explicitly captures information from different parts at multiple spatial scales on the object.

2.3.4 Classification Network

To fully utilize the abundant patch representations, we explore different ways to form a strong combining network. Based on the assumption that only a small fraction of the patches contains discriminative information and patch contribution should therefore be weighted, we explore the following strategies:

**Fixed patch selection**: take the average score for a fixed number of top ranking patches. This strategy can also predict the performance ceiling of our PAIRS representation.

**Dynamic patch selection**: employ the sparsely gated network [42] to dynamically learn a selection function to select a fixed number of patches for each given input.

**Sequential patch weighting**: apply a Long Short Term Memory Network (LSTM) to reweight different patch features in a sequential way.

**Static patch weighting**: learn a Multi-Layer Perceptron network, which essentially applies a non-linear weighting function to combine the information from different patches.

We find surprisingly that the MLP network, though the simplest network architecture, achieves the best accuracy of all the above approaches. Details are included below in Section 2.4.3.

2.4 Experimental Evaluation

We test our algorithm on two datasets, CUB-200-2011 and NABirds. The CUB-200-2011 [46] dataset contains 200 species of birds with 5994 training images and 5794 testing images. The NABirds [45] dataset has 555 common species of birds in North America with a total number of 48,562 images. Class labels and keypoint locations for each image are provided in both datasets.
<table>
<thead>
<tr>
<th></th>
<th>back</th>
<th>beak</th>
<th>belly</th>
<th>breast</th>
<th>crown</th>
<th>forehead</th>
<th>left-eye</th>
<th>left-leg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huang [21]</td>
<td>80.7</td>
<td>89.4</td>
<td>79.4</td>
<td>79.9</td>
<td>89.4</td>
<td>88.5</td>
<td>85.0</td>
<td>75.0</td>
</tr>
<tr>
<td>Zhang [55]</td>
<td>85.6</td>
<td>94.9</td>
<td>81.9</td>
<td>84.5</td>
<td>94.8</td>
<td>96.0</td>
<td>95.7</td>
<td>64.6</td>
</tr>
<tr>
<td>Ours</td>
<td>91.3</td>
<td>96.8</td>
<td>89.0</td>
<td>91.5</td>
<td>96.9</td>
<td>97.6</td>
<td>96.9</td>
<td>80.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>left-wing</th>
<th>nape</th>
<th>right-eye</th>
<th>right-leg</th>
<th>right-wing</th>
<th>tail</th>
<th>throat</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huang [21]</td>
<td>67.0</td>
<td>85.7</td>
<td>86.1</td>
<td>77.5</td>
<td>67.8</td>
<td>76.0</td>
<td>90.8</td>
<td>86.6</td>
</tr>
<tr>
<td>Zhang [55]</td>
<td>67.8</td>
<td>90.7</td>
<td>93.8</td>
<td>64.9</td>
<td>69.3</td>
<td>74.7</td>
<td>94.5</td>
<td>N/A</td>
</tr>
<tr>
<td>Ours</td>
<td>76.8</td>
<td>94.6</td>
<td>97.4</td>
<td>80.3</td>
<td>75.3</td>
<td>83.6</td>
<td>97.4</td>
<td>90.5</td>
</tr>
</tbody>
</table>

Table 2.1: **Pose Estimation (Keypoint Prediction) Results.** Accuracy measured with PCK (Percentage of Correct Keypoints).

2.4.1 Keypoint Prediction Performance

We use PCK (Percentage of Correct Keypoints) score to measure the accuracy of our keypoint prediction approach. A predicted keypoint $p$ is “correct” if its within a small neighborhood of the ground truth location $g$, i.e. if

$$||p - g||_2 \leq c \times \max(h, w)$$

where $c = 0.1$ is the constant factor used previously [21, 55] and $\max(h, w)$ is the longer side of the bounding box.

We evaluate our pose-estimation network on CUB-200-2011 and compare our PCK scores with those of other methods in Table 2.1. We achieve the highest score on all 15 keypoints with a considerable margin. We do especially well on legs and wings where other models struggle to make precise predictions. Visualization results are shown in Figure 2.5.

Although we localize the wings and legs better than components, they are still worst predicted parts for our model. This is caused by significant pose changes as well as the inherent appearance similarity between symmetric parts. We also note that using keypoints to denote the wings isn’t always appropriate – wings are two-dimensional planar parts that cover a relatively large area. Designating a keypoint for the wing can be ill-posed, it’s challenging to decide which point represents the wing location best. In fact, the ground truth keypoint
location of the CUB dataset is the average of five annotators’ results and it’s quite hard for them to reach a consensus.

2.4.2 Patch Classification Network

We adopt the ResNet-50 [20] architecture for the patch classification network due to its high performance and compact GPU footprint, though alternate architectures like VGG and Inception can easily be adapted. We now discuss two considerations which facilitate training. **Symmetry.** For a given object with $n$ keypoints, the total number of patches that can be extracted is

$$\binom{n}{2} = \frac{n(n-1)}{2} = \mathcal{O}(n^2)$$

which increases quadratically with $n$. Most real world objects show some kind of symmetry. Due to the visual similarity inherent in symmetric pairs of keypoints (for example, right and left eyes, wings and feet), we treat each pair as a hybrid keypoint in the patch generation
process. Many real-world objects, like birds, cats, cars, etc. are symmetric in appearance. Based on this observation, we propose to merge the patches for a symmetric pair of keypoints into a hybrid patch category, e.g. left-eye | tail and right-eye | tail can be merged into the hybrid eye | tail pair, with an appropriate flip of one patch.

As a result, the total number of patch classification networks is reduced from 105 to 69 for the CUB dataset; on the NABirds dataset, the number is reduced from 55 to 37.

Visibility. Due to self-occlusion or foreground-occlusion, not all keypoints are visible in the image. Previous works [21] would eliminate patches with invisible keypoints to purify the input data. Contrarily, we find that this would hurt the performance of the patch classifiers. Details for comparison can be found in Figure 2.7a. We believe this degradation is caused by the reduction in training set size. This is a similar finding to [27] that noisy but abundant data consistently outperforms clean but limited-size data. Additionally, the pose-estimation network will make a reasonable guess even if the keypoint is invisible. So during patch classifier training, all keypoints are considered visible by taking the maximally-activated location.

2.4.3 Classification Network

Based on the assumption that image patches should contribute differently to classification, four different strategies are explored and we describe the details in this section.

Fixed patch selection. We assume that only a few patches contain useful information and others are redundant or even act as noise. We propose a fixed patch selection strategy to keep the best $k$ patches. A greedy search algorithm would evaluate all $n \choose k$ combinations for $k \in [1, n]$. The number of patches grows as $n!$ and quickly becomes intractable. We thus employ the beam search [38] algorithm. Instead of greedily searching the whole parameter space, we iteratively consider larger and larger subsets (values of $k$), while only keeping a limited number, $w$, of the best combinations at each iteration. Thus for a given $k$, we use the $w = 100$ best patch sets from iteration $k - 1$, and consider, in turn, the effect of adding
each patch among those not already in a given patch set. After all such expanded sets are considered, the $w$ best sets are retained toward iteration $k + 1$. To explore the potentially optimal performance of fixed patch selection on our pose-aligned patch representation, we also try this beam search on the test set with results shown in Figure 2.7d. Our observation is that without overfitting, the potential of fixed patch selection should be well above 89%, compared to 87.5% for the current state-of-the-art [29]. Simply averaging the predictions of all patches achieves 87.6% accuracy.

**Dynamic patch selection.** One alternative we experiment with is the sparsely gated network [42] for dynamic patch selection. Different from the beam search algorithm which identifies a static set of patches for all input images, the gated network selects different combination of patches depending on the input images. A tiny network is trained to predict weights for each patch and an explicit sparsity constraint is imposed on the weights to only allow $k$ non-zero elements. A Sigmoid layer is added to normalize the weight. The network architecture can be described as,

$$G(x) = \text{softmax}(\text{top-k}(H(x)))$$

where $H(x)$ represents the mapping function from the input image to the patch weights. $G(x)$ is the patch selection function. Different architectures for the tiny network are tried and we find that a simple linear layer works well most of the time. Best accuracy is achieved when $k = 105$. Interestingly when $k=1$, Our selected patch performs worse than the best performing patch found fixed patch selection, implying the gated network’s inability to learn useful information for decision making.

**Sequential patch weighting.** Recurrent neural networks (RNN) specialize in processing sequential data like text and speech. RNNs have been widely adopted as an attention mechanism to focus on different image regions sequentially. We instead employ an RNN for sequential patch weighting, aiming to discover different patches for decision making. We
employ a one-layer Long Short Term Memory (LSTM) network with 512 nodes. Each node has a hidden layer of size 1024. The last output of the sequence is selected as the final output. We obtain 82.7% in this experiment, confirming the effectiveness of the LSTM network.

**Static patch weighting** The final, and as it turns out the most effective method that we tried is the MLP network. Our MLP network contains one hidden layer with 1024 parameters, followed by a batch normalization layer, a ReLU layer, and then the output layer. On CUB our final accuracy is 88.7%, 1.2% higher than the current state-of-the-art result. We combine the keypoint pair patches with single keypoint patches and achieve a new state-of-the-art 89.2% accuracy. We compare our result with several other strong baselines in Table 2.2.

We also test our algorithm on the NABirds dataset with results shown in Table 2.3. Our algorithm attains an accuracy of 87.9%, more than 5% better than the best known result.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Annotations</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50 [20] Baseline</td>
<td></td>
<td>79.2%</td>
</tr>
<tr>
<td>Bilinear CNN (PAMI 2017)</td>
<td>[31]</td>
<td>79.4%</td>
</tr>
<tr>
<td>Pairwise Confusion [11]</td>
<td></td>
<td>82.8%</td>
</tr>
<tr>
<td>PAIRS</td>
<td></td>
<td>87.9%</td>
</tr>
</tbody>
</table>

Table 2.3: **Performance on the NABirds dataset.**
Figure 2.6: **Single Keypoint vs. PAIRS Patches.** We compare the accuracy of patches around keypoint \( k \) (green dots) with PAIRS patches involving keypoint \( k \) (blue dots). The x-axis is the keypoint id and the y-axis is the patch accuracy. Most single keypoint patch are inferior to PAIRS patches in terms of isolated patch accuracy.

### 2.4.4 Additional Study

**Axis-Aligned vs. Pose-Aligned.** In Figure 2.6, we show a comparison of classification accuracy between pose-aligned patches (keypoint pairs) and axis-aligned patches (single keypoints). Axis-aligned patches consistently perform poorly relative to the pose-aligned patches, confirming the effectiveness of our disentangled feature representation.

**Patch Size Study.** One hyper-parameter in our algorithm is the pose-aligned patch size. We tried several size options on the best performing patch and saw that accuracy is generally higher for larger-size patches. We adopted a \( 256 \times 512 \) patch size because our base model is pretrained for this size.

**Choice of Pose Estimation Network.** To test the influence of the pose-estimation network on the proposed algorithm, we train a separate Stacked Hourglass Network (SHN) [37] model for comparison. While the SHN model is about 2% better than the Fully-convolutional Network (FCN) in it’s PCK score, the final classification accuracy numbers are comparable.
Figure 2.7: Results Visualization. (a) We show patch classification network accuracies using two strategies, visible-only and all keypoint patches. This affirms that treating all keypoints as visible improves patch classifier accuracy. (b) Hard case mining by correct prediction patch number with sample images ranging from hard to easy. (c) Distributions of patch classifier performance. Some examples are shown in the text box. (d) Beam search results using two strategies, patch finding on the training (blue) and testing (orange) sets. The latter is purely for the estimating of the potential of the PAIRS representation.

2.4.5 Results Visualization

We show the classification accuracy for each patch when considered independently on the CUB dataset in Figure 2.7c. The best performing patch corresponds to belly | crown, achieving 79.6% accuracy. The worst performing patch is the left-leg | right-leg pair which achieves only 15.7% accuracy. Empirically, global patches perform better in isolation than local patches, however local patches are also very important for localizing discriminative
object parts. The best set of patches found by beam search (see Figure 2.7d) provides insight – a combination of global and local patches are selected to achieve an optimal result.

As hard cases often can only be classified by a few highly-localized discriminative parts, the number of patches with correct predictions reflects the difficulty of the image. We propose to use the fraction of patches correctly predicting the class of an image as an indicator of image difficulty. In Figure 2.7b, a histogram is shown, plotting the number of many images (y-axis) for which only the given number of patches (x-axis) correctly predicted the class. Example images are shown below, ranging from hard on the left, to easy on the right; hard cases can be due to very easily-confused classes or to pose-estimation failure.

2.5 Conclusion

Pose variation constitutes a major challenge in fine-grained recognition, one which recent methods fail to effectively address. This paper introduces a unified object representation built upon pose-aligned patches instead of image-aligned regions – this representation disentangles the intrinsic appearance of an object from confounding influences such as pose variation. Our proposed algorithm attains state-of-the-art performance on two key fine-grained datasets, suggesting the critical importance of disentangling pose and appearance in fine-grained recognition.

Acknowledgements This work was supported by the National Science Foundation under Grant No. IIS1651832. We gratefully acknowledge the support of NVIDIA Corporation for their donation of multiple GPUs that were used in this research.
References


Abstract

Network interpretation as an effort to reveal the features leaned by a network remains largely visualization-based. In this paper, our goal is to tackle semantic network interpretation at both filter and decision level. For filter-level interpretation, we represent the concepts a filter encodes with a probability distribution of visual attributes. The Decision-level interpretation is achieved by textual summarization that generates an explanatory sentence containing clues behind a network’s decision. A Bayesian inference algorithm is proposed to automatically associate filters and network decisions with visual attributes. Human study confirms that the semantic interpretation is a beneficial alternative or complement to visualization methods. We demonstrate the crucial role that semantic network interpretation can play in understanding a network’s failure patterns. More importantly, semantic network interpretation enables a better understanding of the correlation between a model’s performance and its distribution metrics like filter selectivity and concept sparseness.
3.1 Introduction

Network interpretation seeks to illuminate or expose the features that have been learned, and its difficulty lies in the end-to-end learning of the feature extraction and classification sub-networks, which typically contain millions of parameters each. “Debugging” an over-confident network, one which assigns the wrong class label to an image with high confidence, can be extremely difficult, especially when adversarial noise [7] is added to deliberately mislead the network to the wrong conclusion. In that case a meaningful explanation is highly desirable, which contains features responsible for triggering the error, similar to the syntax error highlighting of an intelligent compiler. A thorough understanding of the neural networks is an indispensable part for their continuous success. Network interpretation is also crucial for tasks involving humans due to legal reasons. It is therefore important to distill the knowledge learned by deep models and present it in an easy-to-understand way.

Most popular approaches for network interpretation are visualization-based. Filter-level interpretation (understanding the concepts a filter encodes) is often achieved by displaying the maximally activated dataset example [33] (Figure 3.2) or the optimized input image with image prior regulation [16, 18]. Decision-level interpretation (understanding why the
network makes a decision, also called attribution) [21, 23, 25, 26, 37] is often achieved by highlighting a region in the image that’s important for the decision-making. Despite their success at providing visual clues, pure visualization is unable to provide semantic explanation and sometimes misses detailed information, as shown in Figure 3.1. Similarly, Adebayo, et al. [1] argues that “reliance, solely, on visual assessment can be misleading.”

Humans, on the other hand, can justify their conclusions using natural language. For instance, a knowledgeable person looking at a photograph of a bird might say, “I think this is an Anna’s Hummingbird because it has a straight bill, and a red throat and crown. It’s not a Broad-tailed Hummingbird because the latter lacks the red crown.” This kind of textual description carries rich semantic information and is easily understandable. Semantic information is a logical medium in which to ground the interpretation of deep convolutional models, serving as a beneficial supplement for the visualization methods.

This paper focuses on semantic network interpretation at both filter and decision level. An intuitive way for semantic filter interpretation is to assign a single concept to each filter, as did in [2]. However, the filter-concept relation is usually not one-to-one: a filter can represent several concepts and a concept can be encoded by multiple filters. This distributed characteristic improves a model’s representation efficiency by design [12]. We instead propose to represent a filter with a conditional multinomial probability distribution, called the filter-attribute distribution (see Figure 3.3 for an example). Intuitively, an attribute $t$ is more likely to represent a filter $f$ if images containing $t$ frequently activate filter $f$. We further tackle semantic interpretation for network decision using textual summarization. Textual summarization aims to find a list of visual attributes that the network is basing its decision on. A natural sentence is generated with the top attributes as supporting evidence. A direct application of textual summarization is network debugging, generating descriptive error messages when the network’s prediction is wrong, and it helps us to identify three major failure patterns for the fine-grained dataset CUB-200-2011 [31] (Section 3.4.1).
Figure 3.2: Examples of filter visualization using images with maximal activation, each masked by their corresponding feature maps. One limitation of the visualization-based filter interpretation is the lacking of **diversity**: it is unable to capture the whole space of represented concepts with a limited number of data samples.

We devise a Bayesian inference algorithm to compute the posterior probability that a filter $f$ is activated by a visual attribute $t$ as $p(t|f)$. The difference between our algorithm and a visual attribute prediction algorithm is that the later usually associates visual attributes to an image in a supervised way, but ours associates visual attributes to filters and decisions in an unsupervised way. The goal of network interpretation is not to predict the target label but to loyally reflect the internal working mechanism of a neural network. The key differences between this work and network dissection [2] are that we use a Bayesian algorithm to represent a filter with an attribute distribution instead of a single concept and we only leverage image-level caption annotations.

The filter-attribute distribution provides a tool to quantitatively understand how concepts are encoded by filters. Specifically, we explored the correlation between a model’s performance with the distributed level of its representation. Two metrics of distributed representation are examined, namely filter selectivity and concept sparseness [4]. Filter selectivity is measured by the number of distinctive concepts a filter represents, and concept sparseness refers to the way a single concept is distributed among filters. Understanding the correlation between a network’s performance with its distributed characteristics could potentially lead to new optimization functions to train better networks. Section 3.4.3 provides
a thorough evaluation and discussion. Further more, an ablation study shows that deleting less selective filters is likely to cause more damage to a network, a contrast to our intuition. Human study shows that 41.5% of users think textual attributes are a better medium for network interpretation than visualization, and users find 80.1% of the top 5 attributes in the filter-attribute distributions are accurate.

3.2 Related Work

Network interpretation – Two main approaches to network interpretation exist in the literature: filter-level interpretation [2, 5, 8, 15, 16, 18, 19, 24, 27, 32, 33, 36] and decision-level interpretation (or attribution) [21, 22, 37, 38]. The goal of filter-level interpretation is to understand the features that a specific filter (also known as neurons) learns. While it is easy to directly visualize the first convolutional layer filter weights and understand the patterns they detect, it makes little sense to directly visualize deeper layer filter weights because they act as complex composite functions of lower layers’ output. Early examples of filter-level interpretation include finding the maximally activated input patches [33] and visualizing the guided back propagation gradients [24]. Some works [16] try to synthesize visually pleasant preferred input image of each filter through back-propagation into the image space. [18] applies a generator network to generate images conditioned on maximally activating certain last-layer neurons. The Plug and Play paper [19] further extends [18] to introduce a generalized adversarial learning framework for filter-guided image generation. Network dissection [2] connects each filter with predefined concepts like object, part, color, etc.

Attempts of decision-level interpretation mainly focus on visualizing important image subregions by re-weighting final convolutional layer feature maps. Examples include [21, 23, 25, 26, 37]. However, the visualization based method only provides coarse-level information, and it remains hard to intuitively know what feature or pattern the network has learned to detect. More importantly, the holistic heat map representation is sometimes insufficient to justify why the network favors certain classes over others when the attentional maps for
different classes overlap heavily. See Figure 3.1 for example. [35] proposes represent the image content and structure by knowledge graph.

**Visual attribute prediction and image captioning** – Visual attribute prediction and image captioning [6] are related but fundamentally different tasks to semantic network interpretation. Visual attribute prediction and image captioning are often supervised with the goal to approximate ground truth labels. Semantic interpretation, on the other hand, aims to loyally reflects the knowledge learned by a model in an unsupervised way. There are no ground truth labels to approximate in semantic interpretation.

We note that [10] defines a task similar to ours, to explain and justify a classification model. Their model is learned in a supervised manner, with explanations generated from an LSTM network which only implicitly depends on the internal feature maps. It is essentially an image captioning task that generates captions with more class-discriminative information. Our method is unsupervised and does not rely on another black-box network to generate descriptions.

**Class activation map and network dissection** – Class Activation Map (CAM) identifies the most important region in an image by the linear combination of final conv-layer feature maps, whose weight is from the parameter in the fully connected layer that connects the feature map to the class label: $M_c(x, y) = \sum_k w^c_k f_k(x, y)$, where $M_c(x, y)$ measures the importance of spatial location $(x, y)$ for class $c$. $f_k(x, y)$ is the value at $(x, y)$ on the $k$th filter’s feature map. $w^c_k$ is the weight that connected the $k$th feature map to the prediction class $c$.

Network dissection [2] is perhaps the most similar work to ours. In [2], a filter is associated to a concept by measuring the overlap between the thresholded filter feature map and the concept segmentation mask. Intersection over union $\text{IoU}_{k,c}$ is proposed to represent the accuracy of unit $k$ in detecting concept $c$. The main different between our work and network dissection is that we model the filter | attribute relation as conditional multinomial probability distribution and propose a general Bayesian inference algorithm to
link a filter to multiple attributes. Our algorithm relies only on image-level caption annotation instead of pixel-level segmentation annotation.

Figure 3.3: An example of filter-level semantic interpretation using filter-attribute distribution, which is a probability distribution of visual attributes that best describe the concepts encoded by a filter.

3.3 Bayesian Inference Framework

For the filter-level interpretation, we seek to represent each network filter with its respective activation patterns in terms of visual attributes. Constructing a paired filter \( | \) attribute dataset is unrealistic, because the filter (as a composite function) is not a well-defined concept with concrete examples. Instead, we propose leveraging off-the-shelf image caption annotations because they contain rich textual references to visual concepts. The intuition behind our filter-attribute association is simple: a filter can be represented by the images that strongly activate them and the visual attributes contained in such images should have a high probability of representing the filter. The joint consensus of all images in the dataset can increase the probability of the relevant visual attributes and suppress that of the irrelevant visual attributes.

3.3.1 Filter-Attribute Distribution

We denote \( \mathcal{F} = \{ f_i | i = 1, \ldots, m \} \) as the group of final conv-layer model filters. We denote \( \mathcal{X} = \{ x_j | j = 1, \ldots, n \} \) as the set of input images. The filter \( f \)'s output for input \( x \) is written
as \( f(x) \) (with a slight abuse of notations), which we call a feature map or filter activation.

We consider models \([9, 13]\) with a global pooling layer \( \phi \) followed by a single fully-connected layer. The global pooling layer output for \( x \) is written as \( \phi(f(x)) \). The output of the fully-connected layer is the class prediction from \( \mathcal{C} = \{c_k | k = 1, \ldots, o\} \). The weight matrix of the fully-connected layer is \( W^{o \times m} \). A list of textual attributes from \( \mathcal{T} = \{t_l | l = 1, \ldots, p\} \) is attached to each image. We loosely denote by \( t \in x \) if \( t \) is contained in \( x \)'s attribute list. \( x^t \) represents images that contain attribute \( t \).

We’re interested to know the representative visual attributes for a filter in the network’s final-conv layer. For a given filter \( f \), the probability that an attribute \( t \) can represent its activation pattern is:

\[
p(t|f) \propto p(f|t)p(t) \tag{3.1}
\]

\( p(t) \) is the prior probability for visual attribute \( t \). We consider the relative importance of attributes because attributes carry different amount of information. For example, “small bird” has less information than “orange beak” because the latter appears less in the text corpora and corresponds to a more important image feature. We employ the normalized TF/IDF feature as the attribute prior.

\( p(f|t) \) measures the likelihood of attribute \( t \) activating filter \( f \). As attributes are not directly involved in the neural network, we introduce the input image as a hidden variable:

\[
p(f|t) \propto p(f|\mathcal{X}, t)p(\mathcal{X}, t) = \prod_j p(f|x_j, t)p(x_j, t) \tag{3.2}
\]

where \( \mathcal{X} \) represents the set of input images. \( p(x_j, t) \) measures the probability that \( x_j \) contains \( t \):

\[
p(x_j, t) = \begin{cases} 
1 & \text{if } t \in x_j \\
0 & \text{otherwise.}
\end{cases} \tag{3.3}
\]
Figure 3.4: An overview of the algorithms for decision-level and filter-level semantic interpretation using class-attribute distribution (top row) and filter-attribute distribution (bottom). Visual attributes in every image are weighted by the activation strength and its importance factor (TF/IDF) to generate filter-attribute distribution (section 3.3.1). The filter-attribute distribution are re-weighted by linear layer weight and the activation strength to generate class-attribute distribution (section 3.3.2).

We use $x^t$ to represent images containing $t$ in their attribute list. According to Eqn 3.2 and Eqn 3.3, images without attribute $t$ are zeroed out, so we have $p(f|t) \propto \sum_j p(f|x_j^t, t)$. $p(f|x_j^t, t)$ measures the likelihood that the image $x_j^t$ and the attribute $t$ are the reason for filter $f$’s activation. $f$ is conditionally independent of $t$ given $x_j^t$:

$$p(f|x_j^t, t) = p(f|x_j^t) \propto \phi(f(x_j^t))$$  (3.4)

where $\phi(f(x_j^t))$ is the global pooling layer output for input $x_j^t$ and filter $f$, which measures how likely an image will activate a filter. To summarize, the posterior probability that attribute $t$ is the reason that filter $f$ activates is given by:

$$p(t|f) \propto \text{TF/IDF}(t) \prod_j \phi(f(x_j^t))$$  (3.5)
Figure 3.5: Examples of textual summarization that contains the top visual attributes that are accountable for a network’s decision-making. Note that the goal of textual summarization is not to accurately predict the image attributes, but to loyally reflect the reasons behind a neural network’s classification decisions.

The approximation that $p(f|x^t_j, t) = p(f|x^t_j)$ neglects the fact that when an image activates a filter, the feature map favors certain attributes over others. For example, if $f(x_j)$ highlights the head area of a bird, attributes related to “head”, “beak” or “eyes” should be assigned with higher probabilities than attributes related to “wings” and “feet”. Although this approximation assigns equal probability to all visual attributes inside an image, it actually works quite well in practice, as the joint consensus of all input images boosts true attributes and suppresses false ones. Note that the proposed method can easily adapt to datasets with other forms of annotations like keypoints or part segmentation. Higher probability can be assigned to the visual attributes associated with a part when the feature activation map overlaps highly with its segmentation mask.

3.3.2 Textual Summarization

With the help of the filter-attribute distribution, we can find the top attributes that account for the network’s classification decision. This task can be formulated as the probability that a visual attribute $t$ is the underlying reason given the fact that the network predicts input image $x$ as class $c$. We introduce final convolutional layer filters $\mathcal{F}$ as hidden variables and by marginalizing over $\mathcal{F}$ we get:
\[ p(t|x, c) \propto p(t|\mathcal{F}, x, c)p(\mathcal{F}|x, c) \]
\[ = p(t|\mathcal{F})p(\mathcal{F}|x, c) \]
\[ = \prod_k p(t|f_i)p(f_i|x, c) \tag{3.6} \]

where \( p(t|x, c) \) is the probability that \( t \) is the reason for the network predicting class \( c \) for image \( x \). \( t \) is conditionally independent from \( x \) and \( c \) given \( \mathcal{F} \), such that \( p(t|\mathcal{F}, x, c) = p(t|\mathcal{F}) \). \( p(t|f_i) \) can be computed from filter \( f_i \)'s attribute distribution using Eqn 3.5. \( p(f_i|x, c) \) measures the importance of filter \( f_i \) in the decision-making process, and it’s proportional to the product of the global pooling layer’s output of \( f_i \) denoted as \( \phi(f_i(x)) \) and the weight between filter \( f_i \) and class \( c \), \( w_{i,c} \):

\[ p(f_i|x, c) \propto \phi(f_i(x))w_{i,c} \tag{3.7} \]

We call \( p(\mathcal{T}|\mathcal{X}, \mathcal{C}) \) the class-attribute distribution.

A sentence is generated to describe the network’s decision-making process using the class-attribute distribution. Although it’s popular to employ a recurrent model for sentence generation, our task is to faithfully reflect the internal features learned by the network and introducing another network could result in additional uncertainty. We instead propose a simple template-based method using the top \( n \) attributes, with the following form:

"This is a \{class name\} because it has \{attribute 1\}, \{attribute 2\}, ..., and \{attribute \( n \).""

3.4 Experiments

We evaluate the proposed algorithms on the fine-grained bird dataset of CUB-200-2011 [30] which contains 5997 training images and 5797 testing images. Two ways to obtain image-level attribute annotations for the CUB-200-2011 dataset are explored. The first is to leverage
Figure 3.6: Each row represents a network failure — an incorrectly predicted class label. From left to right, each column shows the query image, canonical images for both the ground-truth and the incorrectly predicted classes, and the textual explanations for each of these classes.

The image caption annotations provided by Zhang, et al. [34], which include five captions for every image that describes the visual features the bird in the image has. Visual attributes are extracted from the captions as adjective-noun word phrases. The CUB-200-2011 dataset also provides visual attribute annotation. There are 312 total attributes to be labelled for each image. Examples include: “Has bill length::longer than head”, “Has back color::grey” and “Has back color::grey”, etc. Although the visual attribute annotation can be more accurate, visual attributes from the captions are more diverse and fine-grained. All the visual attributes shown in this paper are generated from image captions.

To extract visual attributes from the image captions, we follow the process of word tokenization, part-of-speech tagging and noun-phrase chunking. A total of 9649 independent attributes are obtained. The Term Frequency (TF) of phrase $t$ is computed as the number of occurrences of $t$ in the same captioning file. The Inverse Document Frequency (IDF) is calculated as $\log(N/D)$ where $N$ is the the total number of files and $D$ in the number of files containing phrase $t$.

Examples of the generated filter.attribute distribution are shown in Figure 3.3. Examples of the generated textual explanations for image classification are shown in Figure 3.5.
3.4.1 Network Debugging

In figure 3.6, we show three major patterns of network failure through textual summarization. In the first example, a Tree Sparrow is incorrectly recognized as a Chipping Sparrow because the network mistakenly thinks “long tail” is a discriminative feature. According to wikipedia, American Tree Sparrows have a rufous stripe through the eye; on Chipping Sparrows it’s black. Tree sparrows also have a spot in the middle of the breast and a bicolored bill that Chipping Sparrows don’t have. Failing to identify the correct features for discrimination is the most common source of errors across the dataset. In fine-grained classification, the main challenge is to identify discriminative features for visually-similar classes, differences of which are often subtle and localized to small parts.

The second example shows a Seaside Sparrow that has mistakenly been recognized as a Blue Grosbeak. From the textual explanations we ascertain that the low image quality mistakenly activates filters that correspond to blue head and blue crown. The underlying source of this error is complex – the generalization ability of the network is limited such that small perturbations in the image can result in unwanted filter responses. Such failures imply the critical importance of improving network robustness to noisy inputs.

In the third case, the network predicts the image as a Yellow Warbler, however the ground-truth label is Yellow-bellied Flycatcher. According to a bird expert, the network got this correct – the ground-truth label is incorrect! The network correctly identifies the yellow crown and yellow head, both obvious features of the Yellow Warbler. Errors like this are not surprising because, according to [29], the class labels on roughly 4% of the CUB dataset are incorrect.

3.4.2 Human Study

**Visualization vs. semantic interpretation** – We conduct human study using the Amazon Mechanical Turk platform to evaluate the proposed semantic interpretation. Our first study aims to know users’ preference between visualization methods and semantic interpretation.
using textual summarization. "A picture is worth a thousand words". It’s not surprising that normal users prefer the visualization method that have dominated the network attribution field. However, our study shows that 41.5% of users prefer textual explanations and think they provide more helpful information than the visualization methods. This study confirms that semantic interpretation can serve as a helpful alternative or complement to network visualization.

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>No.1</th>
<th>No.2</th>
<th>No.3</th>
<th>No.4</th>
<th>No.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter-attribute distr.</td>
<td>80.1</td>
<td>93.9</td>
<td>92.9</td>
<td>89.8</td>
<td>74.3</td>
<td>53.2</td>
</tr>
<tr>
<td>Class-attribution distr.</td>
<td>75.8</td>
<td>89.2</td>
<td>88.1</td>
<td>83.3</td>
<td>66.4</td>
<td>51.9</td>
</tr>
</tbody>
</table>

Table 3.1: The turkers rated accuracy for filter-attribute distribution and class-attribution distribution.

**Filter-attribute distribution evaluation** – In order to evaluate the filter-attribute distribution, we list the top five attributes along with the top-activated images for each filter. The users are instructed to select the attributes that are present in most, if not all
the highlighted regions of these images. Generally 80.1% of the attributes are regarded as accurate to describe the highlighted regions. Specifically, the accuracy for each of the five attributes are 93.9%, 92.9%, 89.8%, 74.3% and 53.2%. This study shows that most of the visual attributes are rated as relevant to reflect the filter’s activation pattern. To evaluate our textual summarization algorithm, we asked the turkers to rate the top five attributes for each image. The average accuracy was 75.8%. The per-attribute accuracies were 89.2%, 88.1%, 83.3%, 66.4% and 51.9%. Further study reveals another interesting phenomenon: for images that are correctly predicted by the network, 76.3% of our top attributes were regarded as accurate; for incorrectly predicted images the accuracy was 73.8%. This indicates the probability that a neural network learns better features for correct classification than incorrect classification.

3.4.3 Network Understanding

If network interpretation is about knowing what features a network had learned, network understanding cares more about what makes a good feature. Defining good features and finding a way to learn them is crucial for the continuous success of deep models. Bengio, et al. [3] listed several characteristics a good feature representation should have, e.g. disentangling factors of variation, smoothness, abstraction and invariance, and distributed representations. Many efforts [11, 14, 17, 20, 28] have been devoted into understanding each of these properties. In this section, we focus on the distributed representation and its correlation with a model’s performance.

Filter Selectivity – A filter’s selectivity refers to its representing a small number of concepts. A strongly-selective filter only activates on a narrow set of visual attributes. These filters are more interpretable than those whose activation patterns spread widely across many visual attributes. The entropy of the filter-attribute distribution serves as a good indicator of a filter’s selectivity; low entropy means a sparse attribute distribution and strong selectivity. A few questions arise: do models with more interpretable filters perform better?
Figure 3.8: (a) Visual attributes sorted by the number of filters that encodes them. (b) The scatter plot of filter selectivity (inversely proportional to its attribution distribution entropy) and filter importance (inversely proportional to correct sample reduction after removing it from the model).

Is the opposite true? Is there a significant correlation between a model’s performance with its filters’ selectivity?

We compare the filter selectivity of three different models with increasing numbers of parameters: ResNet-18, ResNet-50, and ResNet-152. The classification accuracy on the CUB dataset for these models is 73.2%, 81.6% and 83.4% respectively. The entropy of their filter-attribute distribution are shown in Figure 3.7. Note that ResNet-18 has 512 filters and ResNet-50, ResNet-152 have 2048 filters. The first two graph on the top row of Figure 3.7 show the box plot and sorted distribution entropy for the three models. ResNet-18 has the highest entropy and, by definition, lowest selectivity among all three models. ResNet-50 has more strongly-selective filters than ResNet-152 although the later is more accurate.

To understand how filter selectivity evolves during training, we take four snapshots of a ResNet-18 network during training with epoch number 0, 1, 2 and 50. The box plot and sorted distribution entropy are shown in last two figures on the top row of Figure 3.7. Before training, the network has a lower number of selective filter, but the filter selectivity is not strictly increasing during training.

We next study the correlation between a filter’s importance and its selectivity. A filter’s importance can be measured by the performance drop after deleting it from the
model: filters with higher correct sample reduction are of greater importance. We sequentially remove the final convolutional layer filters, one at a time, and record the decrease in correctly predicted samples. We compare the reduction of correct samples against the filter’s selectivity in the scatter plot shown in Figure 3.8b. Overall, deleting one filter typically has very little impact on the model’s performance (±5), but note that the highest correct sample loss occurs when some of the most weakly-selective filters are deleted. Removing a strongly-selective filter is less likely to result in a performance drop compared to weakly-selective filters. This experiment shows that a filter’s importance is surprisingly negatively related to its selectivity. We hypothesis that an important filter encodes some rare concepts and a less important filter encodes some concepts that are highly duplicate. We find that the concepts represented by the most strongly-selective filter are: “yellow crown, black throat, black cheek patch”, which are encoded by many filters. Deleting such a filter is less likely to cause a significant dip in a model’s performance.

Concept Sparseness – Concept sparseness refers to the fact that a concept is represented by several filters. We represent a concept’s sparseness by the number of filters whose top 10 activation pattern contains such a concept. Figure 3.8a shows the most popular concepts (visual attributes) in descending order. ‘black crown’, the top concept, is spread across 179 filters, followed by the ‘long neck’ concept spread across 149 filters. Note that ‘Black crown’ is also the most frequent attribute in the caption annotation file. The bottom row of Figure 3.7 shows how concepts are encoded in different models and how they changed during the process of training. ResNet-18 has less concepts encoded than ResNet-50, which is then followed by ResNet-152. During the training phrase for each model, the total concepts reduced but the number of filters that encode a concept increases. Generally speaking, better model encodes more concepts and the concepts become increasingly more distributed in the filters during training.
3.5 Conclusion

In this paper, we focus on the task of semantic network interpretation at both filter and decision level. We represent the concepts a filter learns as a conditional multinomial probability distribution on visual attributes. A Bayesian inference algorithm is proposed to compute the attribute distribution for both filters and network decision. We study the correlation between a model’s performance with its distributed representation. Two metrics (filter selectivity and concept sparseness) are examined. Generally, better models have higher filter selectivity and encode more concepts. During training, the filter selectivity increases and the concepts become increasingly more distributed in the filters. For decision-level semantic interpretation, textual summarization is generated to justify a network’s classification results and can be used to uncover the common failure patterns on fine-grained recognition. Human studies are conducted to evaluate the accuracy of the proposed algorithms and validate the importance of semantic network interpretation.
References


Chapter 4

On Global Feature Pooling for Fine-grained Visual Categorization

Abstract

Global feature pooling in modern network architectures remains underexplored in the literature of fine-grained visual categorization. This work presents an empirical and experimental study towards a thorough yet intuitive understanding of and extensive benchmark of popular pooling approaches. A variety of valuable findings emerge. Max pooling is found to encourage networks to learn part-level features and consistently performs better than average pooling. K-max pooling achieves the best accuracy – on par with bilinear pooling at a much lower computational cost. A closer look at generalized pooling reveals that max pooling is less prone to overfitting. Post-global batch normalization is effective for reducing overfitting across pooling schemes. Finally, we explored the integration of heterogeneous pooling methods and found a “freeze-and-train” strategy works best.
Figure 4.1: **Global average vs max pooling.** We visualize the 4 top activated final conv-layer filters using the maximum activated images masked by their feature maps. Average pooling focuses more on object-level features, while max pooling focuses on part-level features.

### 4.1 Introduction

Deeply rooted in works on complex cells in the visual cortex [14] and locally orderless images [20], feature pooling has been an indispensable component of visual recognition in both traditional bag-of-words (BOW) frameworks [2, 22] using hand-crafted features (e.g. SIFT [27], HOG [4]), and in modern convolutional neural networks (CNNs) [21, 23]. A recent variant of this technique, called “global feature pooling” [25], distinguishes itself by defining its pooling kernel to be the same size as the input feature map. The pooling output is a scalar value indicating the presence or absence of certain features (or patterns). Benefits of global pooling are two-fold [25]: they allow for better interpretation of the underlying filters as feature detectors, and they serve as a strong network regularizer to reduce overfitting. Global pooling is thus used in most, if not all, recent state-of-the-art deep models [11–13, 37, 39] in visual recognition. Unless otherwise noted, all the pooling methods discussed in this paper are used as the global pooling layer.
Feature pooling is of special interest for Fine-grained Visual Categorization (FGVC) [6, 30, 31], where objects are classified into subcategories rather than basic categories. Carefully designed pooling schemes can learn helpful discriminative features and yield better performance without stacking additional conv-layers in the network. Examples include [40] which combines three pooling operations (average, max and cross-channel pooling) in order to learn to capture class-specific discriminative patches. Another major research direction is higher-order pooling: [26] proposed to apply bilinear pooling (also know as second-order pooling) to capture pairwise correlations between feature channels and to model part-feature interactions; [8] proposed compact bilinear pooling which applies random Maclaurin projection and tensor sketch projection to approximate the outer product operation, greatly reducing network parameters without sacrificing accuracy. Other works in this line of research include low-rank bilinear pooling [18], Grassmann pooling [41], kernel pooling [3], and alpha-pooling [35]. Higher-order pooling methods output a vector rather than a scalar, but they’re still highly relevant to this discussion as they are functionally analogous to the global pooling layer. To the best of our knowledge, the effectiveness of other global pooling schemes largely remain unexplored in fine-grained recognition.

The most common pooling operations are average, max, and striding. Striding always takes the activation at a fixed location and is never applied as global pooling. An abundant set of pooling flavors exist for both traditional and modern feature extractors. Stochastic pooling [42] randomly chooses an activation according to a multinomial distribution decided by activation strength in the pooling region. Fractional max pooling [9] can be adapted to fractional sized pooling regions. Spatial pyramid pooling [10] outputs the combination of multiple max pooling with different sized pooling kernels. S3Pool [43], (stochastic spatial sampling pooling) randomly picks a sub-region to apply max pooling to. Detail-preserving pooling [32] computes the output as the linear combination of input feature pixels whose weight is proportional to differences of the input intensities. Translation invariant pooling [44] borrowed the idea of anti-alias by low-pass filtering from signal processing. A major pooling
family, generalized pooling, aims to find a smooth transition between average and max pooling: k-max pooling [17] outputs the average of the k highest activations of the feature map; $l_p$ norm pooling generalizes pooling to the $p$-norm of the input feature map [1]; soft pooling [1], or softmax pooling, outputs the sum of feature map weighted by softmax output; mixed pooling [24] computes a weighted sum of the max and average pooling; gated pooling [24] is similar to mixed pooling but the weight is learned instead. To the best of our knowledge, these pooling operations remain largely unexplored in the global pooling scenario.

An interesting observation is that all highly-ranked classification models [11–13, 37, 39] “happen” to choose the same averaging operation in their global pooling layer. Is this an arbitrary choice or actually the optimal strategy? Research has shown [1, 12, 29, 33, 45] that the selection of feature pooling affects the algorithm’s performance, for both hand-crafted and deep features. Murray, et al. [29] showed that max pooling has superior performance in traditional recognition frameworks because of its better pattern discriminability. The same conclusion was made by an experimental evaluation of [33] using LeNet-5 [23] on the Caltech-101 [7] and NORB [16] datasets. Boureau, et al. [1] provided a theoretical proof that “max pooling is particularly well suited to the separation of features that are very sparse.” However, in squeeze and excitation networks [12], global max pooling gets lower accuracy than average pooling. Similar results were reported by [45] using VGG [36] and GoogleNet [38]. It seems that max pooling is less preferred as a global pooling scheme than before. These intriguing contrasts call for a careful examination of both pooling schemes.

Visualizing the final conv-layer feature maps and filters is helpful to fully understand the vital role that pooling plays in feature learning. The facts that max pooling produces sparser final conv-layer feature maps, and that, the highlighted region correlates well with object parts, leads to the conclusion that:

*global average pooling encourages object-level features while global max pooling focuses more on part-level features.*
Table 4.1: Equations for nine popular pooling methods. We denote the input as $x$ and the output as $y$. $N$ is the size of the input feature map. Refer to Section 4.2 for more details.

As class-specific (class-discriminative) features often reside in localized object parts in fine-grained datasets, one might equivalently say that global max pooling finds more discriminative features, which is well aligned with previous findings [29, 45].

To find the optimal pooling scheme across datasets and models, we have evaluated nine representative pooling schemes: average, max, k-max, $l_p$ norm, soft, logavgexp, mixed, gated, and stochastic pooling. These nine pooling methods are shown for convenience in Table 4.1. We found that max pooling outperforms average pooling in several settings – across datasets, input resolutions, and models. The reason behind this phenomenon, besides their feature differences, is that max pooling is less prone to overfitting. Most pooling methods perform better than average pooling, with k-max ($k = 2$) and mixed pooling ($\alpha = 0.5$) being the top two. The k-max pooling model, when trained properly, even beats the both an unmodified second-order pooling model and its memory compact version with the same backbone network (ResNet-50). The computational and memory cost of k-max pooling is much smaller compared to second-order pooling, which involves high-dimensional matrix multiplication. Stochastic pooling works best when used with the Inception backbone. The fact that no single pooling method works best across all models suggests the potential for adaptive/learnable pooling,
where the pooling function is not chosen by heuristic, but optimized via gradient descent. Such a learnable pooling can be static (fixed for all inputs) or dynamic (updated per input). However, we find that when generalized pooling (a pooling with a hyperparameter that provides a smooth transition between max and average pooling) transitions from max to average, the model performance decreases and the gap between training and testing accuracy increases in an almost monotonic way. The pooling that minimizes the training loss does not necessarily perform the best on the testing set. Throughout our experiments, post-global batch normalization is applied as another key ingredient that achieves consistent performance improvement and faster convergence.

Since different features can be learned by average or max pooling, training a model with heterogeneous pooling will lead to better performance. Common strategies like mixed pooling and channel split showed limited improvement. Our hypothesis is that the gradient from different pooling layers interfere with and cancel each other out, when trained together. We propose to apply a “freeze-and-train” strategy. The intuition is that the frozen branch won’t degrade during training and the gradients will be well separated. The resulting architecture adds a negligible amount of parameters to a backbone network, but consistently outperforms single pooling models.

To summarize, the main contributions of this paper include:

- A thorough study on the differences between max and average global pooling through network interpretation and quantitative analysis.
- An experimental evaluation of nine representative pooling schemes with insightful findings.
- An investigation in the heterogeneous pooling integration showing that the “freeze-and-train” strategy works best.
4.2 Pooling Schemes Overview

Here we describe in detail the pooling algorithms we evaluate in the experiment section. We use the following notations: the input feature map is a 3D Tensor: \( X \in \mathbb{R}^{c\times h\times w} \), with \( c, h, w \) being the channel size, height and width. Each feature map is a 2D matrix, but in most cases the algorithm cares less about the 2D structure than individual activation strength, so we simply use its flattened vector form \( \mathbf{x} \). The scalar output of each feature map is denoted as \( y \).

We use \( x_i \) to index individual elements of \( \mathbf{x} \) and set \( N = hw \) as the feature map size.

**Average pooling** can be expressed as:

\[
y = \frac{1}{N} \sum_{i=1}^{N} x_i
\]

(4.1)

**Max pooling** picks the maximum value of \( \mathbf{x} \):

\[
y = x_i, i = \operatorname{argmax}(\mathbf{x})
\]

(4.2)

**Stochastic pooling** chooses a single activation from the input based on a multinomial distribution decided by relative activation strength.

\[
y = x_i, i \sim P(p_1, ..., p_i, ..., p_N),
\]

(4.3)

where each probability \( p_i \) is given by \( p_i = \frac{x_i}{\sum_{j=1}^{N} x_j} \). We can safely assume \( x_i > 0 \) holds for all \( i \) as the input is always the output of a ReLU layer.

**\( L_p \) norm pooling** computes the \( L_p \) norm:

\[
y = \left( \frac{1}{N} \sum_{i} x_i^p \right)^{\frac{1}{p}}
\]

(4.4)

When \( p = 1 \), \( L_p \) norm reduces to average pooling and when \( p \to \infty \), \( L_p \) norm approaches max pooling.
**Soft pooling**, or softmax pooling, reweights the inputs as:

\[
y = \sum_i \frac{\exp(\beta x_i)}{\sum_j \exp(\beta x_j)} x_i
\]  
(4.5)

When \( \beta = 0 \), soft pooling equals average pooling, and when \( \beta \to \infty \), soft pooling behaves like max pooling.

**Logavgexp pooling** applies three functions consecutively to the input:

\[
y = \frac{1}{\beta} \log \frac{1}{N} \sum_i \exp(\beta x_i)
\]  
(4.6)

Logavgexp is average pooling when \( \beta = 0 \) and max pooling when \( \beta \to \infty \).

**K-max pooling** averages the top \( k \) activation:

\[
y = \frac{1}{K} \sum_i x_i, i \in \text{argmax}_k(x)
\]  
(4.7)

With a slight abuse of notation, we use \( \text{argmax}_K \) to return the indices of the top \( k \) activations. When \( k = 1 \), k-max pooling is the same as max pooling, and when \( k = N \), k-max pooling is average pooling.

**Mixed pooling** combines max and average pooling:

\[
y = \alpha f_{\max}(x) + (1 - \alpha) f_{\avg}(x)
\]  
(4.8)

When \( \alpha = 0 \), mixed pooling is average pooling, and when \( \alpha = 1 \), mixed pooling becomes max pooling. Note \( \alpha \) is not learned in this case.

**Gated pooling** learns a universal weight for combining max and average pooling:

\[
y = \sigma(w^T x) f_{\max}(x) + (1 - \sigma(w^T x)) f_{\avg}(x),
\]  
(4.9)

where \( \sigma(w^T x) = \frac{1}{1+\exp(-w^T x)} \) is the sigmoid function.
4.3 A Comparison of Max and Average Pooling

This section discusses the interpretation and integration of features learned by max and average pooling. The difference between features of max and average pooling is demonstrated through feature maps and filter visualization. Two perceptually-consistent sparsity measurements are proposed. We also discuss ways to integrate two heterogeneous pooling schemes.

**Difference in Feature Maps** - The feature maps generated as input to max-pooling are visually much sparser, with sharply peaked activations. Features maps from the average-pooling model, on the other hand, are more widely activated, as shown in Figure 4.1.

**Difference in Filters** - The images causing the highest activations of a given filter are selected and their feature maps are superimposed as indicators of the attentional region. Figure 4.1 shows examples of filter visualization: the global average pooling features are mostly object-level with nearby/adjacent background patterns, e.g. white birds, with sky, grass, and/or water. Global max pooling, on the other hand, focuses on highly-localized object parts, e.g. eyes, legs, wing bars. In fine-grained datasets, discriminative features often reside in localized parts. Global max pooling is thus able to find highly discriminative, class-specific features, which aligns well with previous findings [1, 29].

**Quantitative Analysis** - The canonical way to measure feature map sparsity is through its $l_0$-norm or entropy. The $l_0$-norm measures the number of non-zero elements of an input and the entropy is used to quantify the “uncertainty” of an input. However, their results are not always consistent with humans’ perception of sparsity. A tiny perturbation in the feature map that is imperceptible to humans may cause big differences in $l_0$-norm and sparsity values. For example, a grayscale feature map (pixel value ranges between 0-255) filled mostly with “1”s is perceptually sparse because “1” is close to pure black, but it has a very high $l_0$-norm and thus is considered “dense”. Adding random noise to a feature map whose magnitude is within ±5 will cause dramatic difference in the entropy value, when it is hardly noticeable to human eyes.
To better reflect perceptual sparsity and allow for better noise tolerance, we propose two modified versions of the above metrics, denoted as $S^{(1,2)}$. Both proposed metrics work by averaging the sparsity of an individual feature map, denoted $s^{(1,2)}$, across $n$ channels and $m$ images: 

$$S^{(1,2)} = \frac{1}{nm} \sum_{j=0}^{n} \sum_{i=0}^{m} s_{ij}^{(1,2)}$$

The first metric is **discrete entropy**: values in a feature map are discretized into a normalized histogram with $k$ bins. The height of each bin is $p_i$, and $\sum_i p_i = 1$. Discrete entropy preserves most of the information in a probability distribution and is less sensitive to small perturbations. The discrete entropy $s^1$ of a feature map is computed as: 

$$s^1 = \sum_{i=0}^{k} -p_i \log(p_i)$$

The second metric is **thresholded $l_0$-norm**. Consider each feature map as a 1D-vector. Let the feature map value at location $i$ be $x_i$ and the feature map size be $N$. When the threshold is $t$, the thresholded $l_0$-norm $s^2$ for each feature map is defined as: 

$$s^2 = \sum_{i=0}^{N} 1(x_i \geq t)$$

where $1(x_i \geq t)$ is an indicator function that returns 1 if $x_i \geq t$ and 0 otherwise. Small positive values are suppressed by thresholded $l_0$-norm.

**Heterogeneous Pooling Integration** - A simple baseline for heterogeneous pooling integration is to treat CNNs as feature extractors [34]. Features from max and average pooling models are concatenated and are fed into an offline classifier. This method is easy to implement but it doubles the model complexity and is not end-to-end learnable.

The first end-to-end learnable solution for heterogeneous pooling integration is mixed pooling [24], which combines the weighted output of the two pooling functions (see Sec. 4.2 for details). Another integration method is called “channel split”, which splits the final conv-layer channels into two halves and applies one type of pooling to each half. The third approach is a “branching” strategy that adds an additional classification branches containing a linear layer on top of a pooling layer [1, 10, 40]. The above methods adds only a tiny amount of parameters to the model. Our hypothesis is that heterogeneous pooling layers help to learn diverse features and benefit the model’s performance, but our experiments show that the above methods have only tiny performance gain over single-pooling models. One possible reason is that lower layer filters become confused by the gradient signals coming
from the two different pooling schemes. We propose a simple modification to the branching strategy to separate the gradient flow of different pooling methods using a “freeze-and-train” strategy: the backbone model with a global average pooling layer is trained until convergence; their weights are then frozen when a new linear layer with global max pooling is added. The newly added linear layer is trained until convergence. The whole network is trained using a smaller learning rate as the final step. The “freeze-and-train” strategy achieves consistent improvement over single-pooling models and the experimental results are provided in Section 4.4.3.

<table>
<thead>
<tr>
<th>Pooling Method</th>
<th>Architecture and Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ResNet-50</td>
</tr>
<tr>
<td></td>
<td>Birds</td>
</tr>
<tr>
<td>Average</td>
<td>84.96±0.32</td>
</tr>
<tr>
<td>K-max</td>
<td>k = 2</td>
</tr>
<tr>
<td>Mixed</td>
<td>α = 0.5</td>
</tr>
<tr>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>LogAvgExp</td>
<td>β = 1</td>
</tr>
<tr>
<td>(L_p)</td>
<td>(p = 2)</td>
</tr>
<tr>
<td>Gated</td>
<td></td>
</tr>
<tr>
<td>Soft</td>
<td>β = 1</td>
</tr>
<tr>
<td>Stochastic</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Mean accuracy and standard deviation are shown for nine pooling schemes on three fine-grained datasets using ResNet-50. VGG-19, Inception-v3 and ResNext-50 are used as backbones on the birds dataset. The best accuracy on each dataset is in bold and the second best is underlined. The best performers are: k-max, mixed, and max pooling.
4.4 Experiments

The experiment section covers three major topics: the benchmarking and analysis of popular pooling schemes, the quantitative analysis of feature map sparsity, and the integration of heterogeneous pooling.

Three public fine-grained datasets are used in the experiments: CUB-200-2011 (Birds) with 200 classes, 5,994 images for training and 5,794 for testing; Stanford Cars (Cars) with 196 classes, 8,144 images for training and 8,041 for testing; and FGVC Aircraft (Aircraft) with 100 classes, 6,667 images for training and 3,333 for testing. ResNet-50, VGG-19, Inception-v3 and ResNext-50 are used as backbone networks in the pooling benchmark experiment. We use stochastic gradient descent (SGD) with universal learning rate schedule for ResNet-50, VGG-19 and ResNext-50 models, and the Adam optimizer [19] for Inception-v3. ResNet-50, VGG-19 and ResNext-50 models are trained with a learning rate of $10^{-3}$ for 30 epochs, and $10^{-4}$ for 20 more epochs, in most cases. Inception-v3 models are trained with a learning rate of $10^{-4}$ for 30 epochs and $10^{-5}$ for 20 epochs. All models converged under the described settings after fine-tuning from pretrained weights on Imagenet [5]. Batch size is 16, weight decay is $10^{-4}$ and momentum is 0.9. The input image size is $448 \times 448$ in most cases, aligned with previous works [3, 26].
4.4.1 Global Pooling Evaluation

This section presents a detailed evaluation of nine global pooling schemes on three widely-used fine-grained datasets. We discuss how hyperparameters influence generalized pooling and what these experiments imply for the possibility of learnable generalized pooling.

Five of the nine pooling methods each contain one hyperparameter. They are $p \in [1, \infty)$ for $l_p$-norm pooling; $\beta \in [0, \infty)$ for soft pooling; $\beta \in [0, \infty)$ for logavgexp pooling; $k \in \{1, 2, ..., N\}$ for k-max pooling, and $\alpha \in [0, 1]$ for mixed pooling. For the pooling benchmark experiment, we select a canonical value for each hyperparameter for simplicity: namely $p = 2, \beta = 1, \beta = 1, k = 2, \alpha = 0.5$. These hyperparameters are chosen for simplicity: most of them are the smallest integers in their feasible parameter range ($\alpha$ in mixed pooling simply chooses the average of its max and min values).

The global pooling benchmark results are shown in Table 4.2. The left three columns show results produced with ResNet-50 backbones, the last three columns show model results using VGG-19, Inception-v3 and ResNext-50 as backbone networks. The top-performing method on each dataset is highlighted in bold and the second best is underlined. On ResNet-50 backbone, k-max pooling ($k = 2$) has the best performance among the nine pooling methods we tested, with the highest accuracy across all three datasets, attaining 1.34%, 0.63%, and 0.99% increases over the average pooling baseline for Birds, Cars and Aircraft, respectively. These results are actually highly competitive considering the difficulty of fine-grained datasets. If we train k-max pooling ($k = 2$) longer ($10^{-3}$ for 50 epochs, $10^{-4}$ for 30 epochs and $10^{-5}$ for 20 epochs), it can reach 87.20% accuracy on the Birds dataset, surpassing all higher-order pooling methods proposed for fine-grained recognition (e.g. 86.4% from kernel pooling [3], 85.3% from alpha pooling [35]). K-max pooling also has very low memory and computation complexity when compared to the higher-order pooling methods.

The second best method is mixed pooling ($\alpha = 0.5$). It obtained the third best result on the Birds dataset and second best on Cars and Aircraft. It is another highly competitive pooling method that has been neglected in fine-grained recognition, though it shows larger
variance than other algorithms. Max pooling is the third best overall performer, more or less on par with mixed pooling. Nearly all of the generalized pooling methods outperform the average pooling baseline. Soft pooling is the worst performer, with several entries below baseline. The overall worst performing method is stochastic pooling, with the lowest accuracy for all three datasets.

On the right side of Table 4.2 we show results on the Birds dataset using VGG-19, Inception-v3 and ResNext-50. The purpose of this experiment is not to optimize for performance, but to compare pooling methods between different backbone models. Max, mixed and k-max pooling still perform strongly on different backbones. Although stochastic pooling attains the best accuracy (83.32%) on Inception-v3, the overall accuracy using Inception-v3 is inferior to all other backbones. Additionally, stochastic pooling is among the worst performers on backbones other than Inception-v3 and ResNext-50.

These results show clearly that average pooling is not an optimal choice for fine-grained recognition. Our careful examination of multiple pooling methods supports our intuition about the importance of localized features for fine-grained recognition, and these features aren’t captured very well when using global average pooling.

**Image Resolution** Table 4.2 shows that max pooling outperforms average pooling across datasets and models. These results were obtained using $448 \times 448$ input images. To test robustness to image size, we trained 10 models on inputs ranging from $224 \times 224$ to $448 \times 448$. Results are shown on the right of Figure 4.2. Max pooling outperforms average pooling by a non-trivial margin for all tested resolutions.

Taken together, the above results illustrate our first observation:

1. **Max pooling performs better than average pooling when fine-tuned from a pretrained model for fine-grained visual categorization, across various datasets, models and input resolutions.**

**Generalized Pooling** Table 4.2 only includes results for one value of each hyperparameter for the generalized pooling methods. We next evaluate the performance of three generalized
pooling methods with varied parameters. For k-max pooling, we train 196 models, one for each possible value of $k$. For $l_p$-norm and mixed pooling, we trained 10 and 20 models with linearly spaced parameters. Figure 4.3 shows the training and testing accuracy. For all three cases, although small perturbations exist, generalized pooling seems to be bounded by the extremes of max and average pooling. As average pooling transforms to max pooling (as $k$ decreases from 196 to 1, $\alpha$ from 0 to 1 and $p$ from 1 to 100), the mean accuracy increases nearly monotonically, with few exceptions. The generalization gap (the gap between training and testing accuracies) in all three graphs clearly show that max pooling always has smaller gap than average pooling and is by definition less prone to overfitting.

From these phenomena we make the following observations:

2. Performance increases almost monotonically as average pooling moves to max pooling.
3. Max pooling outperforms average pooling by reducing overfitting.

**Learnable Generalized Pooling** Our models so far have all involved human-designed pooling, sometimes with a hand-selected hyperparameter. It would be preferred if the pooling function could be *learned*. Generalized pooling provides an intuitive route: a smooth transition between average and max pooling can be achieved if the network learns the optimal hyperparameter. The hope is to learn a “magic” pooling parameter that outperforms both
average and max pooling. But learning such a generalized pooling function is complicated by the fact that performance appears to be bounded by max pooling; and more importantly, the learned pooling – trained to minimize training loss – is likely to converge to average pooling, which has the lowest training loss and highest training accuracy, thus leading to overfitting as shown in Figure 4.3.

**Post-global Batch Normalization**  Batch normalization [15] has been adopted by most popular models to reduce output shift and to help with convergence. It is common practice to add batch normalization to the output of convolutional layers. We find, however, that adding a batch normalization layer to the output of the global pooling layer provides a universal improvement for all models we’ve tested. It is shown that dependence on “single directions” (single filter outputs) is an indicator of the network’s overfitting [28] and we find in our experiments that the global pooling output actually has a very large variance between filters, indicating a risk for emphasizing single directions. The advantage of post-global batch normalization is two-fold: it helps models with different pooling methods to converge, and it universally improves the performance by $\sim 1\%$. This margin, coupled with Figure 4.3, indicates that reducing overfitting may be an important topic in fine-grained recognition.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average</th>
<th>Max</th>
<th>MLP</th>
<th>Mixed</th>
<th>Split</th>
<th>Branching</th>
<th>Freeze</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birds</td>
<td>84.96</td>
<td>86.11</td>
<td><strong>86.66</strong></td>
<td>85.97</td>
<td>85.49</td>
<td>85.24</td>
<td>86.40</td>
</tr>
<tr>
<td>Cars</td>
<td>92.29</td>
<td>92.84</td>
<td><strong>93.25</strong></td>
<td>92.88</td>
<td>92.56</td>
<td>91.90</td>
<td><strong>93.08</strong></td>
</tr>
<tr>
<td>Aircraft</td>
<td>88.55</td>
<td>89.14</td>
<td><strong>89.62</strong></td>
<td>89.38</td>
<td>88.45</td>
<td>88.72</td>
<td><strong>89.97</strong></td>
</tr>
</tbody>
</table>

Table 4.3: **Heterogeneous pooling integration.** The offline training method (MLP) generally performs best. Freeze-and-train works best among the end-to-end learnable models.

**4.4.2 Perceptual Sparsity for K-max Pooling**

This section evaluates how perceptual sparsity metrics change with respect to different $k$ values in k-max pooling. For an input of size $448 \times 448$, the final feature map size is $14 \times 14 = 196$, and thus $k \in \{1, \ldots, 196\}$. 16 evenly spaced points are sampled $\{1, 14, 27, \ldots, 196\}$; recall
that k-max pooling is max pooling when k=1 and average pooling when k=196. The Birds dataset is used for training. Results are shown in Figure 4.2.

Histogram bin number is a hyperparameter for discrete entropy. We report results for multiple bin numbers \{4, 8, 16\}. Max pooling (k=1) gets 0.018, 0.095, 0.34 and average pooling (k=196) gets 0.10, 0.40, 0.91 respectively. The gaps between them are 0.082, 0.305, 0.57. For thresholded \(l_0\)-norm, results are reported using threshold values as: 0.01, 0.015, and 0.02. Max pooling (k=1) gets roughly 4, 3 and 1 and average pooling (k=196) gets roughly 15, 8, 6. The gaps are approximately 11, 5, and 5.

As lower discrete entropy and thresholded \(l_0\)-norm means higher sparsity, it is validated that max pooling encourages sparser final-conv layer feature maps and the sparsity increases monotonically as average pooling is transformed to max pooling.

### 4.4.3 Heterogeneous Pooling Integration

In this section, we describe the experiments for heterogeneous pooling integration. We use ResNet-50 as the backbone model. In the offline classifier experiment, we first trained two models with global average and max pooling separately. The two models are then applied to each image as a feature extractor. We repeat this process four times on the training set with varied random seeds as dataset enhancement. A multi-layer perceptron (MLP) is trained using the concatenated features. The MLP we used has 4096 input nodes and 1024 hidden nodes. The number of output nodes equals the number of classes. The learning rate is \(10^{-4}\) for 30 epochs. The offline MLP classifier gets 86.66\%, 93.25\% and 89.62\% respectively on the three fine-grained datasets, generally surpassing all of the single training stage pooling models, the only exception being freeze-and-train on the Aircraft dataset (see Table 4.3).

For the channel splitting experiment, we use 0.5 as a splitting ratio: half of the feature maps are fed into average pooling and the other half into max pooling. The mean accuracies are 85.49\%, 92.56\% and 88.45\%. The branching strategy gets 85.24\%, 91.90\% and 88.72\%. Both strategies are worse than max pooling. Mixed pooling achieves 85.97\%, 92.88\% and
89.38%, as shown in Table 4.2. Mixed pooling is slightly better than max pooling on two datasets, and worse than max pooling on one.

For the freeze-and-train method, we modify the branching strategy in the following way: the backbone with global average pooling is trained for 50 epochs and then frozen. The newly added linear layer with global max pooling is then trained for 80 epochs before the whole network is trained for 2 more epochs. This strategy guarantees that the final performance won’t degrade when the new pooling layer is added. As a result, the final accuracies on the three datasets are 86.40%, 93.08% and 89.97%, surpassing all end-to-end learnable methods and even the offline MLP in the case of the Aircraft dataset.

4.5 Conclusion

In this paper, we focus on the detailed analysis of the global pooling layer in popular classification models, applied specifically to the task of fine-grained recognition. We found that max pooling produces much sparser feature maps and helps the network learn part-level features. Average pooling, on the other hand, encourages object-level features to be learned. We evaluated nine representative global pooling schemes for fine-grained recognition. K-max ($k = 2$) pooling outperformed all other global pooling schemes and is actually better than bilinear pooling and compact bilinear pooling models. Max pooling performs better than average pooling across datasets, models, and input resolutions. Max pooling generalizes better (exhibits a smaller train-test gap) than average pooling. Model performance exhibits a monotonically increasing characteristic as generalized pooling is transformed from favoring average towards favoring max pooling. We highlight the importance of post-global batch normalization – which is absent from most, if not all, popular state-of-the-art models – in attaining faster convergence and in consistently improving model performance. The freeze-and-train strategy performs best among all end-to-end learnable models for heterogeneous pooling integration.
References


[41] Xing Wei, Yue Zhang, Yihong Gong, Jiawei Zhang, and Nanning Zheng. Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification. In ECCV, 2018.


Chapter 5

LMPNet for Weakly-supervised Keypoint Discovery

Abstract

In this work, we explore the task of semantic object keypoint discovery weakly-supervised by only category labels. This is achieved by transforming discriminatively-trained intermediate layer filters into keypoint detectors. We begin by identifying three preferred characteristics of keypoint detectors: (i) spatially sparse activations, (ii) consistency and (iii) diversity. Instead of relying on hand-crafted loss terms, a novel computationally-efficient leaky max pooling (LMP) layer is proposed to explicitly encourage final conv-layer filters to learn “non-repeatable local patterns” that are well aligned with object keypoints. Informed by visualizations, a simple yet effective selection strategy is proposed to ensure consistent filter activations and attention mask-out is then applied to force the network to distribute its attention to the whole object instead of just the most discriminative region. For the final keypoint prediction, a learnable clustering layer is proposed to group keypoint proposals into keypoint predictions. The final model, named LMPNet, is highly interpretable in that it directly manipulates network filters to detect predefined concepts. Our experiments show that LMPNet can (i) automatically discover semantic keypoints that are robust to object pose and (ii) achieves strong prediction accuracy comparable to a supervised pose estimation model.
5.1 Introduction

Although according to Aristotle, “the whole is greater than the sum of its parts”, representing an object as a collection of features from its part constellation is proven to be more powerful [1, 12, 41] than holistic object-level representations. Part-based representations have been widely adopted by tasks like classical object detection [1, 12], fine-grained visual categorization [6, 41], and person re-identification [24], etc. As an indispensable component in the part-based representation, object keypoint localization has been well studied in previous works [8, 29, 37]. Different from supervised algorithms whose reliance on manual annotation may hinder their application to large-scale datasets, we pursue a different route for keypoint discovery, weakly-supervised by easier-to-obtain object category labels. Beside its obvious benefits of reduced annotation cost, weakly-supervised keypoint discovery has its own virtue that enables us to manipulate and transform discriminatively-trained filters to learn semantic features as desired using carefully-designed network components.

Weakly-supervised learning [2, 9, 18, 31, 35, 40, 43, 45, 46] has gained popularity in the research community over the last few years. Representative tasks include object detection [9, 35, 45], semantic segmentation [2, 40], instance segmentation [18, 46] and dense correspondence [31, 43] using images labels or captions. Among the extensive research efforts, one particular interesting direction is to leverage the aggregated discriminatively-trained intermediate network filters. We argue that the key to success for filter aggregation is to identify the desired characteristics of the target output and transform the filters accordingly. Although network filters have been shown to learn to detect different concepts during discriminative training [3, 44, 45], there is a lack of a systematic ways to manipulate the learned internal representations to be driven by specific needs. Previous algorithms “guide” the filter learning process by injecting hand-crafted loss terms [31, 43]. However, these losses are only indirectly influencing the network and their relative weights are hyper-parameters that are tricky to balance. In this paper, we propose several novel network components that impose direct constraints on internal network filters and feature maps based on solid evidence
The proposed leaky max pooling layer helps the network learn “non-repeatable local patterns” that align well with object keypoints. Each $2 \times 2$ region represents one final-conv layer filter with activations. The detected object keypoints include bird beak, eyes, feet and wing tip; vehicle logo, rear-view camera, wheels and frontal light.

that is from neural network interpretation, and our efforts, in return, contributes to a deeper network understanding.

Work from Choe, et al. [10] suggested that weakly-supervised object localization can be ill-posed, as network filters can learn to represent both objects and background, and the aggregation of filter activations is not guaranteed to be within the object boundary. We further argue that weakly-supervised keypoint discovery can be in vain without proper definition of object keypoints. In this paper, a semantic object keypoint is defined as a non-repeatable local pattern. Examples of semantic keypoints include: landmarks on the human face (eyes, nose, mouth, etc.), the eyes, beaks and feet of a bird, or the logo and headlights of a vehicle. Keypoint-level features are spatially smaller than part-level features, examples of which are the cheek, forehead, wings and belly of a bird, and the windshield and doors of a vehicle. Background regions (sky, lake, trees, etc.) and object-level properties (body color, the pose of flying or perching) are not regarded as keypoints.
Figure 5.2: **LMPNet Architecture.** The final conv-layer feature maps are fed into two branches: a classification branch with leaky max pooling and a keypoint prediction branch with selection and learnable clustering. The most discriminative region (the first keypoint prediction) is then masked out from the input image and fed into a replica network.

Next, we show how the definition of “non-repeatable local patterns” leads naturally to the development of a novel global pooling [25] layer. Under this definition, keypoint detectors should produce spatially sparse feature maps similar to a narrow spatial Gaussian distribution. One key insight to draw is the influence of the global pooling layer on final conv-layer feature map sparsity. Max pooling has been shown to produce sparser feature maps compared to its average pooling counterpart, but the max operation is not optimal as it doesn’t prevent a pattern existing everywhere, violating the “non-repeatable” rule. An ideal pooling layer should detect a local pattern that only exists at the maximum activation location. We thus propose a novel pooling layer, named leaky max pooling, that not only allows one pixel out like max pooling, but also suppresses other pixels with negative weights. Leaky max pooling can be implemented by reformulating the pooling operation as a matrix-vector multiplication, for which average and max pooling are shown to be special cases. Only a tiny amount of computation is introduced between both the forward and the backward processes. Leaky max
pooling explicitly encourages sparse activations that are well-aligned with object keypoints. It is a plug-and-play module that can be applied to most modern network architectures.

It can be tempting to assume that filters develop into “perfect” keypoint detectors with leaky max pooling, however, two phenomena were observed that challenge this assumption. First, filters do not necessarily consistently activate on a tight cluster of patterns. A significant number of filters actually activate on multiple patterns (a mechanism known as distributed representation). Blindly aggregating filters would lead to poor keypoint predictions. After extensive investigation of the filter visualizations, we find a strong correlation between a filter’s activation strength and its selectivity. A higher activation more likely indicates the existence of the certain underlying patterns. A simple yet effective selection strategy is therefore proposed that only keeps highly-selective filters. Second, keypoint proposals are not evenly distributed across the object. A majority of filters end up detecting the same concept – the most discriminative feature – like the head of a bird. This causes difficulty for the keypoint prediction as keypoints from less discriminative parts are poorly represented in the keypoint proposals. A key observation is that if the most important image region is masked out, the network will be forced to diversity its attention to other parts.

The feature maps before the learnable clustering layer are named “keypoint proposals”, which potentially contain object keypoints. To produce final keypoint predictions from the keypoint proposals, a many-to-one projection is learned where predictions are decided by votes from multiple proposals. The weight between a keypoint proposal and a keypoint prediction gets updated iteratively to reflect their association relationship: keypoint proposals closer to a prediction are assigned larger weights. A greedy strategy is applied to sequentially output keypoint predictions. In each iteration, the most prominent keypoint prediction is produced and proposals that vote for it are erased before the next iteration.

The main contributions of this paper are:

• Identifying three preferred characteristics for keypoint detectors that are missing in discriminatively-trained filters via network visualization.
• Proposing several novel network components that are highly integratable and computationally efficient, including leaky max pooling and learnable clustering, that detect and cluster “non-repeatable local patterns”.

• The proposed network architecture, LMPNet, produces diverse and meaningful object keypoints across different classes, and poses, achieving accuracies comparable to supervised keypoint prediction models.

5.2 Related Work

Lying between supervised [15, 23] and unsupervised learning [4, 11], weakly-supervised learning [31, 32, 46] presents unique challenges to the research community. Although supervised learning remains the dominating scheme for major vision tasks, its demand for human labels often tampers its application to the enormous amount of unlabelled datasets. Unsupervised learning automatically learns features from raw image data, but its training goal is tricky to define. Evaluation of unsupervised learning relies on the definition of good feature representations [5] like disentanglement, but whether it can be learned without inductive bias or is fundamentally necessary are still under debate [16, 26, 28]. Recently, self-supervised learning [13, 20, 21, 30] gains momentum as it generates training signals from the image itself, by predicting patches’ relative location [30] or the rotation angle [13], etc. The dilemma of unsupervised/self-supervised learning is that they are often used to pretrain a model and strong labels are still needed to for downstream tasks like classification [13, 30] and landmark prediction [19, 36].

Weakly-supervised object detection (WSOD) WSOD aims to predict the object bounding box using only image labels. One of the popular models in WSOD is multiple instance learning (MIL) [27]. In MIL, a positive bag contains at least one positive sample and a negative bag contains all negative samples. The standard MIL pipeline includes the repeating process of detector training and object relocating. In recent years, inspired by network visualization, many algorithms [9, 31, 45, 46] has been proposed to aggregate the
feature maps of a discriminatively-trained CNN model to get the object location. Class activation map (CAM) [45] works by reweighting final conv-layer feature maps using the linear layer weights for WSOD. Choe et al. [10] conducted an extensive evaluation of WSOD algorithms and found that WSOD can be a ill-posed task with only image labels as supervision. As the only constraint on network filters is the cross entropy loss for classification, they tend to focus on the most discriminative part, or learn background features instead of object features. Aggregation of these filters may lead to incomplete or over-complete bounding boxes.

**Weakly-supervised part discovery** Early works in fine-grained recognition [22, 39, 42] explores object part detection free of part or keypoint annotation. Krause et al. [22] proposed to generate object mask by co-segmentation and find object parts by aligning the masks. Xiao et al. [39] performs spectral clustering on the intermediate filters to produce filter groups as part detectors. Zhang et al. [42] instead picks filters that generates top responses as part candidates and trains them by iterative negative sample mining. Simon and Rodner [34] proposed an EM-like optimization process to generate a star shape model for a subset of selected part proposals from the internal representation of CNNs. Compared to our end-to-end trainable LMPNet, the above methods usually involve offline process of filter aggregation and selection.

**Weakly-/self-supervised landmark detection** Perhaps the most similar works to ours are AnchorNet [31] and [43]. A set of filter of diversity, discriminality, and sparsity constraint are learned by the proposed loss terms. etc. A series of works utilize the equivariance constraint of 2D image transformation (translation, rotation, scaling) to learn landmarks automatically, which is similar to the concept of self-supervised learning. To evaluate the generated landmarks, a weak network (three layer MLP) is usually used to regress to the ground truth keypoints. Different from the above methods, our algorithm employs no extra loss terms but proposes novel network components that explicitly transform network filters
Figure 5.3: **Examples of different pooling methods.** We show activations for dense and sparse input using average, max and leaky max pooling separately. Leaky max pooling produces low output for dense inputs and high output for sparse inputs.

into keypoint detectors. Moreover, we employ greedy matching PCK to evaluate our keypoint prediction, without supervised training.

### 5.3 LMPnet Architecture

A comprehensive overview of the proposed LMPNet architecture can be found in Figure 5.2. The LMPNet builds on a classification network, replacing the global pooling layer with leaky max pooling. A keypoint prediction branch is added that takes the final conv-layer feature maps as input and produces keypoint predictions. In the keypoint prediction branch, input filters are selected and fed into the learnable clustering layer. The original image with the most discriminative region erased is then fed into a replica network that shares the same architecture but with independent weights. The keypoint predictions from the original network and the replica network are fused as the final predictions. The following sections contain detailed descriptions of the proposed LMPNet network modules.
5.3.1 Leaky Max Pooling

We motivate the leaky max pooling layer by the following toy example (Figure ?? (a)). Suppose there are two $2 \times 2$ feature maps: one feature map is densely activated containing all 1s; the other one is sparsely activated containing all 0s but one 1. An ideal pooling layer should produce high output for sparse feature maps and low output for dense feature maps. Let’s first examine the two most common pooling types: average and max. The average pooling outputs $1/4$ for the sparse input and 1 for the dense inputs. This is undesirable as average pooling encourages dense inputs rather than sparse inputs. Max pooling outputs 1 for both sparse and dense inputs. Compared to average pooling, max pooling produces higher output for sparse inputs, but does not discourage dense inputs. The proposed leaky max pooling produces $1 - 3\epsilon$ for the dense input, lower than both average and max pooling, and 1 for the sparse input, the same with max but higher than average. Therefore, leaky max pooling encourages sparse inputs and suppresses dense inputs at the same time.

Next we show how the global pooling layer can be implemented by a general matrix-vector multiplication operation. Suppose the input to the pooling layer is $X \in \mathbb{R}^{b \times c \times h \times w}$ where $b, c, h, w$ are batch size, channel number, feature map height and width separately. The output of pooling layer is $Y \in \mathbb{R}^{b \times c}$. The input can be reshaped as a 2D matrix for convenience: $\tilde{X} \in \mathbb{R}^{bc \times hw}$. A pooling vector is defined as $w \in \mathbb{R}^{hw}$. The pooling operation can then be reformulated as matrix-vector multiplication: $\tilde{Y} = \tilde{X}w$, and $\tilde{Y} \in \mathbb{R}^{bc \times 1}$ is then reshaped to $b \times c$. Different pooling operations can be distinguished by their respective pooling vector $w$ under such formulation. It is obvious that for average pooling,

$$w_i = \frac{1}{hw}, \quad \forall i, \quad (5.1)$$

and for max pooling:

$$w_i = \begin{cases} 1 & \text{if } i = \text{argmax}(x) \\ 0 & \text{otherwise.} \end{cases} \quad (5.2)$$
Figure 5.4: **Network filter visualization.** Each column represent a filter. Each row represents a group of feature maps that activate a filter. Activation strength decreases from top to bottom and from left to right. We select filters by their activation strength to eliminate noisy filters.

The leaky max pooling is defined by:

$$w_i = \begin{cases} 
1 & \text{if } i = \arg\max(x) \\
-\epsilon & \text{otherwise.}
\end{cases} \quad (5.3)$$

where $\epsilon$ is a small positive number. The above operation introduces only a tiny amount of additional computation. Take ResNet-50 for example, the computation overhead (measured in FLOPS) introduced by the matrix-vector multiplication adds only 0.05% to the whole network.

The gradient distribution of the final conv-layer feature maps provides an even better perspective of leaky max pooling’s advantage. Each element inside the feature map gets a
gradient signal proportional to the corresponding weight of the pooling kernel. For average pooling, every feature map pixel gets the same gradient, positively encouraging patterns to exist across the whole image. For max pooling, only the maximum pixel gets non-zero gradient, encouraging a pattern to exist in a local neighborhood. However, all other pixels get zero gradient, meaning they are neither encouraged nor discouraged. They tend to remain unchanged during the subsequent training iterations. This, a pattern that universally exists, has a high probability of being learned by max pooling. The proposed leaky max pooling overcomes the limits of both the average and max pooling methods by promoting only “non-repeatable local patterns”. Compared to max pooling, non-maximum elements get a negative gradient and are suppressed in the next iteration. The leaky max pooling kernel can also be thought of as a template that matches only sparsely-activated feature maps.

The final conv-layer feature maps before the leaky max pooling layer are spatially sparse and well-aligned with ground truth keypoints, but it is problematic to feed them directly to the learnable clustering layer without proper post processing. These feature maps are noisy, as a filter does not necessarily activates on one specific pattern. Moreover, high-level filters often concentrate on the most discriminative region, leading to unevenly distributed keypoint proposals. In the next two sections, we discuss these issues in detail and describe our methods to tackle them.

5.3.2 Filter Consistency

Network interpretation [3, 44, 45] via visualization has been well studied in discriminative training. Bau et al. [3] further conducted statistical studies of the interpretability of discriminatively-trained network filters and found that not all units are interpretable. For example, only 71% of filters are deemed “interpretable” in AlexNet. They also found that a network’s performance seems to be independent of its filters’ interpretability. In fact, Hinton [17] argued that the distributed representation, where each neuron is responsible for multiple concepts, is what makes a neural network representation powerful. In our case, a
large portion of filters only learn random/noisy features; these filters are the ones with smaller maximum activation strength. This lead us to believe that the network is easily content with its filters as long as they’re discriminative. The least-selective filters learn slower and never get a strong enough gradient to update themselves, winding up with activations on random features.

Another side of the story is that even filters that activate on one specific pattern are not consistent. Our network visualization method works by ranking the output feature maps by their max value. Compared to optimization methods, visualization via dataset samples provides a more realistic and diverse view of a filter’s learned pattern; only samples with top activations are typically shown due to space limitations. However, the whole picture of a filter’s learned pattern can only be revealed by displaying less-activated feature maps and images.

To facilitate a comprehensive understanding of the filter-concept relationship, we design a “full-scale” filter visualization graph shown in Figure 5.4. $2 \times 2$ image tiles are displayed in a 2D grid. Each column represents a filter. Tiles of images in each column are ranked by their images’ activation strength, descending from top to bottom. From left to right, filters are ranked by decreasing maximum activation strength. We sample the filters and images/feature maps evenly along each dimension. It can be seen from Figure 5.4 that images with higher activations are more likely to contain the underlying pattern (the top left portion of the graph). This suggests that a filter’s selectivity (its ability to focus on one specific pattern) is positively related to its activation strength. Intuitively, a higher activation indicates the filter is more confident of the existence of a pattern. We thus propose to use activation strength as a strong clue for filter selection. We simply keep the filters with the highest activations for each input image.
Algorithm 1: Learnable Clustering.

Input : \( X \in \mathbb{R}^{c \times h \times w} \);
Output : \( Y \in \mathbb{R}^{k \times h \times w} \);
Parameter : \( W \in \mathbb{R}^{c \times k} \);
\( W \leftarrow 1 \);
for \( i \leftarrow 1 \) to \( k \) do
    for \( it \leftarrow 1 \) to \( n \) do
        \( w \leftarrow \text{softmax}(W[:,i]) \);
        \( Y \leftarrow w^T X \);
        \( d \leftarrow \text{dist}(Y, X) \);
        \( W[:,i] \leftarrow w + 1/d \);
    end
    \( w \leftarrow \text{softmax}(W[:,i]) \);
    \( Y \leftarrow w^T X \);
    \( Y[i] \leftarrow Y \);
    \( X[d<\text{thr}] \leftarrow 0 \);
    if \( \max(X) = 0 \) then
        break;
    end
end

5.3.3 Filter Diversity

Unsupervised clustering works best on balanced data where each cluster contains a similar number of data samples. However, filter visualization suggests that discriminatively-trained filters tend to focus on the most informative image region. We propose to tackle this issue using attention mask-out. Mask-out can be used for important image region attribution: tiny image regions are sequentially masked out before the image is passed into the network and the classification result is observed. The regions that cause incorrect classification results are regarded as the attention regions of the network. Our idea is that when the most important region is masked out from the input, the network has to learn to distinguish between categories using other regions, thus distributing its attention to other parts of the object.

Our attention mask-out strategy consists of two LMPNet with the same architecture, but with independent weights. The first LMPNet takes the original input and outputs initial keypoint proposals. A binary mask is generated by reversing the first proposed keypoint. The
binary mask is multiplied with the original image to remove its most discriminative region. The masked image is then fed into the second, replica LMPNet while training. During the testing stage, the outputs of both networks are fused to get the final keypoint predictions.

### 5.3.4 Learnable Clustering

We propose a learnable clustering layer to aggregate the keypoint proposals, after selection and attention mask-out, into keypoint predictions. We describe our learnable clustering layer in Algorithm 1. Without loss of generality, we consider the feature map matrix for a single input $X \in \mathbb{R}^{c \times h \times w}$, where $c, h, w$ are channel number and feature map height and width. We set the maximum value of each feature map to 1 and everything else to 0 to remove noise. The layer output is $Y \in \mathbb{R}^{k \times h \times w}$, where $k$ is the number of keypoint predictions. We define a weight matrix $W \in \mathbb{R}^{c \times k}$. The weight matrix elements are initialized to 1. The forward procedure includes $n$ iterations. In each iteration the keypoint predictions are generated by weighting over keypoint proposals using $W$. The distance between keypoint proposals $X$ and keypoint prediction $Y$ is computed using $\text{dist}()$ as the distance between their max value positions. The similarity between $X$ and $Y$ is the reciprocal of their distance. $W$ is updated to reflect the accumulated similarity between the input and output and to account for the contributions of each keypoint proposal to the keypoint prediction. Keypoint proposals that are closer the prediction get larger weights and contribute more to the final result. An erasing step $X[d < thr] \leftarrow 0$ is performed after each keypoint prediction so that keypoint proposals that belong to previous predictions have no influence on new predictions. This is a similar idea to non-maximum suppression [7, 33] in object detection, where object proposals with the highest score are sequentially selected and overlapping proposals are removed. These iterations are repeated until no proposals remain or until all $k$ keypoints are found.
Figure 5.5: **Feature map sparsity comparison.** From top to bottom are filter visualizations obtained using average, max and leaky max pooling. Average pooling learns object-level and background features, max pooling encourages part-level features, and leaky-max pooling focuses on keypoint-level features. See appendix for more visualization results.

5.4 Experiment

This section contains analysis of the leaky max pooling layer and evaluation of the keypoint prediction results. The fine-grained classification dataset CUB-200-2011 [38] is used in these experiments. ResNet-50 [15] is used as the backbone model.

5.4.1 Leaky Max Pooling Analysis

In this section we conduct a thorough analysis of the leaky max pooling layer. We first qualitatively evaluate how leaky max pooling leads to sparse feature maps. Figure ?? (b) shows typical filter visualization for ResNet-50 models using average, max and leaky max pooling separately. Average pooling learns background and object-level features, max pooling learns
part-level features and leaky max pooling learns keypoint level features. A more complete filter visualization can be found in the appendix section. For a quantitative evaluation of the feature map sparsity, we take the normalized final conv-layer feature maps and compute their average entropy: sparser feature maps have smaller entropy. For each pooling method, we show in Figure 5.6 the average feature map entropy for all images in the test set. It is clear that leaky max pooling > max pooling > average pooling in terms of feature map sparsity.

<table>
<thead>
<tr>
<th></th>
<th>AVG</th>
<th>MAX</th>
<th>LMP (ε = 0.1)</th>
<th>LMP (ε = 0.01)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>85.0</td>
<td>86.1</td>
<td>81.4</td>
<td>85.3</td>
</tr>
</tbody>
</table>

Table 5.1: Classification Accuracy for different ε

In leaky max pooling, ε is a hyperparameter that controls how much penalty is imposed on non-maximum locations. We study how different values of ε lead to different results. First we compare the classification accuracy using two different ε values 0.1 and 0.01. These results, together with average and max pooling results, are shown in Table 5.1. By visualizing the filters learned by ε = 0.1 and ε = 0.01, we find the when ε = 0.01, there are a significantly higher number of filters that learn random features than when ε = 0.1. Although ε = 0.01 has a higher accuracy, we choose ε = 0.1 in all our experiments for its better keypoint detection ability. When ε is larger, leaky max pooling is better at suppressing dense inputs.
5.4.2 Keypoint Prediction Evaluation

In this section, we evaluate how well our keypoint prediction corresponds to ground truth keypoints. The final conv-layer feature maps are small, either $7 \times 7$ or $14 \times 14$, depending on the input size. We set the number of final keypoints to $k = 5$ for the purpose of better visualization. We show our final keypoint prediction results in Figure 5.7. All of the predicted keypoints are within the object boundary and are well-aligned with head, tail, feet, wing tips, etc. Our method handles complex bird pose surprisingly well. Examples can be found in the first row of Figure 5.7 that shows our keypoint prediction results for flying birds, a setting where other methods typically struggle.

We next consider a quantitative evaluation of the generated keypoint prediction. Inspired by the traditional PCK (percentage of correct keypoint) metric for supervised
keypoint prediction, we adopt a greedy matching strategy where a keypoint prediction is regarded as accurate if it is within a small range ($\alpha = 0.1$ of the shorter image side length) of a ground truth keypoint. We call this metric greedy PCK. The erasing threshold in our learnable clustering algorithm determines how many keypoint proposals are erased after one iteration of keypoint prediction. We find that setting a smaller threshold generally leads to higher greedy PCK, but keypoint predictions may overlap with each other when the threshold is too small. We show how the greedy PCK varies for different erasing thresholds in Table 5.2. We choose $thr = 3$ in our final model to balance keypoint duplication and keypoint quality.

<table>
<thead>
<tr>
<th></th>
<th>KP 1</th>
<th>KP 2</th>
<th>KP 3</th>
<th>KP 4</th>
<th>KP 5</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>$thr = 3$</td>
<td>96.2</td>
<td>89.2</td>
<td>86.6</td>
<td>82.4</td>
<td>75.4</td>
<td>86.0</td>
</tr>
<tr>
<td>$thr = 2$</td>
<td>97.5</td>
<td>89.1</td>
<td>87.8</td>
<td>83.1</td>
<td>74.8</td>
<td>86.5</td>
</tr>
<tr>
<td>$thr = 1$</td>
<td>96.5</td>
<td>91.4</td>
<td>91.4</td>
<td>90.7</td>
<td>87.6</td>
<td>91.5</td>
</tr>
<tr>
<td>FCN [14]</td>
<td>97.4</td>
<td>96.8</td>
<td>91.3</td>
<td>80.3</td>
<td>75.3</td>
<td>88.2</td>
</tr>
</tbody>
</table>

Table 5.2: **Greedy PCK @ $\alpha = 0.1$ for different erasing threshold.** We choose $thr = 3$ to avoid keypoint duplication and maintain high keypoint prediction accuracy. FCN denotes the fully convolutional neural network result [14] and the 5 keypoints are evenly sampled from their original 15 keypoint predictions.

### 5.5 Conclusion

We propose a novel network architecture named LMPNet for weakly-supervised keypoint discovery using only image class labels as supervision. Novel network components are proposed to explicitly transform discriminatively-trained intermediate network filters into keypoint detectors. Due to the lack of ground truth keypoint annotations, we define semantic object keypoints as “non-repeatable local patterns”, which leads to a novel leaky max pooling layer derived from a generalized global pooling formulation. The pooling vector contains 1 for the max input location and $-\epsilon$ everywhere else. The proposed leaky max pooling layer encourages the final conv-layer filters to have sparse activations that are well aligned with object keypoints. We show by visualization that discriminatively-trained filters have two issues: consistency and diversity. We first propose a simple yet effective selection strategy...
to ensure that filter responses are consistent, and second propose attention mask-out to
distribute the network’s attention to less-represented regions. A learnable clustering layer is
applied to aggregate the keypoint proposals into keypoint predictions, in a sequentially greedy
way. The proposed network architecture is shown to produce meaningful object keypoints
with only image labels. LMPNet works surprisingly well on complex bird poses and achieves
high greedy matching PCK comparable to strongly-supervised baseline models.
References


Chapter 6

Fine-grained Visual Categorization (FGVC): A Comprehensive 10-year Survey

Abstract

Fine-grained visual categorization (FGVC) is a branch of visual object recognition that aims to distinguish sub-classes within a basic class. Examples include species recognition for birds and dogs, and the classification of vehicles’ make, model and year. FGVC presents a unique challenge to object recognition, like its small inter-class variation, large intra-class variation. In recent years, FGVC has become one of the key areas for innovation in object recognition. In this comprehensive survey paper, we present a summary of 200+ papers that are organized into common themes, including foreground segmentation, human-in-the-loop, part-based models, features and pooling, attention and hierarchy/taxonomy. This paper also covers benchmarks on fine-grained datasets, relevant conference workshops and FGVC.org, an online platform that keeps track of the cutting-edge information in FGVC.
6.1 Introduction

Fine-grained Visual Categorization (FGVC), commonly referred to as Fine-grained Recognition, has emerged as one of the key areas for innovation in object recognition in recent years. These innovations include advances in deep neural network architectures, joint solutions for multiple vision problems, and the creation of large-scale fine-grained datasets. As a research area, it started gathering momentum [3] just prior to the deep-learning revolution. In the ensuing years, FGVC has both benefited from and contributed to advances in deep-learning research. The challenges inherent in recognizing objects in fine-grained domains have driven researchers to design innovative approaches which jointly solve such diverse vision problems as pose estimation, segmentation and attribute prediction. Another direction of progress in object recognition, fueled by fine-grained research, is the creation of increasingly large-scale, fine-grained image datasets (e.g. birds [183], airplanes [124], cars [98], and iNaturalist [180]), typically with annotations like object bounding box, segmentation masks, and keypoint, etc.

What we in the computer vision community now refer to as Fine-grained Visual Categorization has long been known as Subordinate Categorization among cognitive psychologists. Rosch, et al. [153] were among the first to outline a categorization spectrum, which ranges from superordinate categories (e.g. animals, vehicles) to basic-level categories (e.g. birds, cars) to subordinate categories (e.g. Peregrine Falcon, Toyota Camry). Superordinate categories differ dramatically in function and purpose. Within a superordinate category such as animals, the different basic-level categories will share similar function and purpose, but will often differ particularly in their morphological (shape) composition. Cats and birds, for example, are composed of very different parts. Within a basic-level category such as birds, the different subordinate categories typically share a common morphological structure or set of parts, but often differ in the properties [153] of those parts such as relative proportions among parts and especially in parts’ appearance. Section 6.2.1 provides a detailed discussion of relevant work on categorization from the psychology literature.
Challenges

Among the difficulties inherent in FGVC, the overarching challenge stems from low inter class variation and high intra class variation. Fine-grained domains typically have very low inter class variation – many subcategories exhibit very subtle and often highly-localized differences in morphology and/or appearance (Figure 1.1). A domain expert is often needed to discriminate between very similar species that, to an average observer, appear to be identical. In fact, analysis [44] of the confusion matrix from multiple algorithms on the ImageNet dataset indicates “a correlation between the structure of the semantic hierarchy (by WordNet) and visual confusion between the categories.”

On the other hand, there can be substantial variation within a given subcategory (intra class variation). Birds provide a prime example. A single species can exhibit significant variation in appearance depending on gender, age/maturity and even season. Males and females often differ in their appearance; males tend to be more colorful and flashy in appearance to draw attention during courtship, while females generally have more drab and nondescript plumage to help conceal them while nesting. Juvenile birds often display a distinct plumage relative to their adult counterparts, with some birds such as gulls donning unique plumages each year until reaching adulthood. Further, many species have different plumages during their breeding season and varying degrees of transitional plumage can be observed before and after.

Approaches

Among the numerous methods proposed for FGVC, we identify six popular themes: foreground segmentation, human-in-the-loop, part-based models, features and pooling, attention, and hierarchy/taxonomy. Foreground Segmentation methods, as the name suggests, segment the object from the background before performing feature extraction. Human-in-the-loop methods harness the power of both computer algorithms and human annotators to form a hybrid learning system. Part-based models represent an object as the concatenation of
Figure 6.1: Themes Over Time - from top to bottom are: foreground segmentation, part-based models, human-in-the-loop, hierarchy/taxonomy, features and pooling, and attention.

(often pose-normalized) part features. Feature and pooling methods aim to find a better feature representation and/or pooling strategy. Attention mechanisms focus sequentially on the most informative regions of an image for better discriminality. Hierarchical/taxonomic methods exploit the “knowledge” from a class taxonomy for better classification. Figure 6.1 shows how the paper distribution of themes has changed over time. For example, foreground segmentation and human-in-the-loop methods were common primarily before the deep learning wave, whereas attention models picked up momentum in later years. Beyond the changes in popular research themes, we also see clear transitions of research interests from hand-crafted features to deep features, from learning multiple models to a single end-to-end learnable model, and from strong supervision with additional annotations to weak supervision with only image labels.
What’s Missing?

Fine-grained recognition has witnessed a dramatic boost in both paper quality and quantity in recent years; however several key components are missing, which casts a potential shadow to its future development. Most papers lack a good interpretation and understanding of the learned model, which is admittedly a common problem because of the “black-box” nature of most deep models. Many papers do provide qualitative visualizations showing their networks’ attentional region, but more quantitative analysis is often missing to clearly validate their claims. Additionally, model complexity is often neglected when reporting results. Some papers unfairly compare their methods to those with weaker backbone models. Most papers report only a single percentage number as the algorithm accuracy, instead of the more rigorous statistics such as average accuracy and standard deviation over multiple runs/trials. Finally, a per-category accuracy analysis would be helpful for better understanding the strengths and weaknesses of a proposed algorithm.

This survey is organized as follows. Section 2 contains background research for FGVC. Section 3 gives a comprehensive list of the representative algorithms proposed, grouped by theme approach. Section 4 covers commonly-used fine-grained datasets and benchmarking/evaluation. We introduce previous FGVC workshops and the fgvc.org website in Section 5. Section 6 provides concluding remarks.

6.2 Background and Related Areas

Fine-grained visual categorization (FGVC) is deeply rooted in pioneering cognitive science research of object recognition and categorization [153]. Based on different levels of abstraction, the categorization problem can be classified into superordinate, basic, and subordinate level categories. Details can be found in Section 6.2.1. It has been found that increasingly detailed levels of categorization require higher degree of perceptual expertise (see Section 6.2.2). We cover early works on object recognition including morphology and shape (in Section 6.2.3) to instance recognition, where animal biometrics plays an important role (in Section 6.2.4). In
recent years, deep learning has dramatically changed the landscape of FGVC, and conversely, FGVC has also contributed novel ideas to deep learning (see Section 6.2.5).

6.2.1 Categorization Spectrum

Categorization of the real world is not arbitrary, but rather organized semantically into taxonomy [153], a system where categories are related to one another by means of class inclusion. Under such a definition, basic categories (car, tables etc.) are at the most inclusive level where all attributes are common to all or most members of the category. Categories one step more abstract are superordinate categories (furniture, vehicle, etc.) whose members share only a few attributes among each other. Categories below the basic level are subordinate categories, which contain many attributes overlapped with other categories.

Two concepts influence the notions of categorization levels in a taxonomy: “cue validity” [153] and “category resemblance” [177]. Cue validity is a probabilistic concept. The validity $p(y|x)$ of a given cue $x$ as a predictor of a given category $y$ increases as the frequency with which cue $x$ is associated with category $y$ increases and decreases as the frequency with which cue $x$ is associated with categories other than $y$ increases. Category resemblance is defined as the weighted sum of the measures of all of the common features within a category minus that of all of the distinctive features. Based on the above definitions, subordinate categories have lower total cue validity and lower category resemblance than do basic categories, as they share most attributes with contrasting subordinate categories [152].

In a follow on work, Tversky, et al. [178] highlighted the role of object parts in categorization. “Parts, more than attributes, distinguish one basic-level entity from another, but members of subordinate categories share parts and differ from one another on other attributes (like part morphology, etc.).” Murphy, et al. [135] suggested, likewise, that subordinate categories generally indicate changes in the features from those usually expected in the basic category, but they preserve the general parts and functions from their basic category.
6.2.2 Perceptual Expertise

A prominent direction in cognitive science research on subordinate categorization is to reflect the behavioral and psychological differences between experts and novices. It was verified [126, 130] that children acquire the ability to categorize subordinate-level concepts only after superordinate and basis-level concepts, presumably because “subordinate categories have a higher between-class similarity, members of different sub-categories look very similar.” Johnson, et al. [90] made an interesting point that throughout the continuum of expertise, conceptual knowledge interacts with perception. Accordingly, experts attend to different and more subtle perceptual features than do novices.

A series of works from Tanak, et al. [172–174] found that expert knowledge is primarily organized at the subordinate level of abstraction rather than at the basic level [173]. For example, a bird expert may spontaneously identify a bird with its subordinate-level name, e.g. robin, instead of its basic-level name, e.g. bird. Although both novices and experts can recognize non-face objects holistically, only the experts seem to be acutely aware of configural properties that distinguish one object from another [174]. Objects in well-learned categories are neurologically differentiated from objects in lesser-known categories at a relatively early stage of visual processing [172]. Focusing on the subject of fish recognition, Boster, et al. [18] reported that expert fishermen judge similarities among fish on both functional and morphological criteria, while novices judge on morphological criteria alone and thereby align with the scientific classification of fish more closely than experts do. Experts also vary more than do novices, presumably because they develop over time, distinct kinds of knowledge on which to base a similarity judgment.

6.2.3 Object Morphology and Recognition

Object recognition research diverted from the pure cognitive and psychology field as a computational task that gave birth to modern computer vision in the early 1970s. Early work
from Minsky, Marr, Biederman, among others, focus on the general frameworks for effective representation of objects and the recognition task is formulated as database searching.

Minsky [131] proposed a general theory for representing visual signals. In its simplest form, the theory states: “when one encounters a new situation one selects from memory a structure called a frame, which is a remembered framework to be adapted to fit reality by changing details as necessary.” For visual scene analysis, the different frames of a system describe the scene from different viewpoints, and the transformations between one frame and another represent the effects of the “changing deltas” (moving from place to place). The frame-systems are linked by an information retrieval network. Once a frame is proposed to represent a situation, a matching process then tries to assign values to each frame’s terminals.

Marr and Nishihara [128] proposed to understand the human visual process by examining its related computational problem. They studied the problem of representing three-dimensional shapes for the purpose of recognition. Specifically, a representation should satisfy the three criteria of accessibility, scope and uniqueness, and stability. The advantages of object-centred coordinate system, volumetric (three-dimensional) primitives, modular organization over a viewer-centred coordinate system, surface-based (two-dimensional) primitives and naive non-modular organization were discussed. Given this representation, shape recognition is a process of associating a newly-derived description with a stored description. Following this line of research, Dickinson, et al. [47] presented an approach to recover and recognise 3D objects from a single 2D image.

Biederman [16] extended the theories of [17, 127, 176] with his “Recognition by Components” theory. Biederman proposed that objects are composed of geometric primitives called geons (short for “geometric icons”) and that the unique spatial configuration of these geons constitutes object identity. Perception of these geon configurations, and in turn of the objects they represent, relies on the existence of non-accidental properties (NAPs) – for example, configurations of image contours that are highly unlikely to have occurred by chance.
6.2.4 Instance Recognition and Biometrics

Instance recognition and biometrics are closely related to subordinate categorization. Both endeavor to discriminate, not between objects from different categories, but between distinct items or individuals in the same category – and to do so based on measurements of observable inherent biological traits or characteristics. While there exist non-visual cues such as voices [32] and writing styles [82], we are primarily interested in unique visually recognizable traits such as gait, faces, fingerprints, and irises. While there has been a tremendous amount of work on recognizing individual humans via faces [187, 228] and other human biometrics (see surveys [20, 88, 125, 170]), relatively little works has been done on animal biometrics. There are a number of researchers and research groups who work closely with zoologists on a variety of animal biometrics problems. Burghardt’s dissertation [30] contains an excellent overview of this extremely challenging problem. Santosh Kumar, et al. have contributed both a thorough survey article [103] and have also published a book [104]. Other key contributions include the Hotspotter approach of Crall, et al. [40] and the Sloop system of Duyck, et al. [53].

6.2.5 Deep Learning

Deep learning [81, 107], or the use of deep neural networks, has been dominating almost all computer vision tasks since the 2012 introduction of AlexNet by Krizhevsky, et al. [101], a transformative application of convolutional neural networks that dates back at least to the LeNet-5 network by Lecun, et al. [106] in late 1990s. The vision community has witnessed a huge shift of interests from the traditional bag-of-visual-words framework with hand-crafted features like SIFT, HOG, etc. to automatically-learned features. A comprehensive survey of deep learning techniques is beyond the scope of this paper and readers are referred to Ian GoodFellow’s Deep Learning book [73] for further details. Unlike previous FGVC survey papers [226] which only covered deep learning methods, this survey is more comprehensive, including both traditional works using hand-crafted features and a broad set of deep learning-based FGVC papers.
The first ripples from the deep learning revolution reached fine-grained recognition in late 2013/early 2014 with the publication of DeCAF [49]. In the ensuing years, fine-grained recognition has both benefited from and contributed to the success of deep learning models in other research domains. For example, attention mechanisms, first introduced in natural language processing, were soon applied in FGVC to sequentially find discriminative image patches. Innovations in FGVC like compact bilinear pooling, kernel convolution and part-based models have been successfully adopted by other research domains like general image categorization, image captioning, and person re-identification etc.

6.3 Themes and Strategies

This survey covers over 200 papers from popular vision-related conference proceedings and journals. To effectively organize them, we divide the papers by common themes among their proposed methods. This section aims to make a comprehensive review of all representative works, in chronological order, for each theme. Readers are referred to “Survey” [226] for an alternative review of deep learning in FGVC. Note that many works do not fall into a single theme or strategy, so each is listed in the subsection for the approach that is most prominent. Papers that belong to none of the common themes are organized in the Misc section.
6.3.1 Segmentation

When objects from different categories are inherently similar in appearance, as they are in FGVC, the appearance of the background across a set of images often has greater variation than that of objects in different objects. Classification algorithms trained to distinguish between the categories therefore often learn the features of the scenes observed instead of the objects themselves. When this occurs, these algorithms are very ineffective at recognizing the objects in new scenes.

One way to overcome this difficulty is to separate the object to be recognized from the background scenery. This task of isolating the object from its surroundings is called segmentation [167]. The process is sometimes referred to specifically as figure-ground segmentation or as segmenting an foreground object (the figure) from the background or clutter (the ground).

Several techniques for FGVC either rely on or leverage segmentation for effective classification, most often using a general-purpose segmentation method such as GrabCut [154]. GrabCut begins with an initial set of foreground and background regions, either manually indicated by a user or more frequently selected automatically, e.g. everything inside a centered rectangle is assumed to be foreground and everything outside is background. These initial regions are used to learn pixel color distributions for both the foreground object and the background. Pixels in the image are then assigned to the foreground or background based on which color distribution they match more closely. The newly predicted foreground and background pixels can be used to refine the color distributions and in turn, relabel all of the pixels as foreground or background. GrabCut’s success relies on the assumption that the pixels in the foreground object are similar to one another, and, simultaneously, dissimilar to those in the background. While GrabCut originally proposed operations at the pixel level, in practice, most segmentation algorithms operate on larger groups of pixels, often called superpixels [149].
Some papers leverage a slightly different figure-ground segmentation algorithm called co-segmentation. Co-segmentation \cite{35, 91, 155} refers to methods that separate the same (or similar) object(s) from their differing backgrounds in two or more images. A generative model is constructed for each image to group foreground and background pixels using a Markov Random Field (MRF). Additional constraints are imposed to penalize differences between the foreground segmentations or to encourage the similarity between them.

Chai, et al. proposed the BiCos \cite{34} (Bi-level Co-Segmentation) algorithm, one of the first segmentation-based technologies, which initially segments images individually and then jointly optimizes all of the individual image segmentations within a class or category together. First, a GrabCut segmentation is performed on each image using a centered rectangle (with half the image’s area) to seed the foreground distribution, and the outer half of the image to seed the background. Next, a super-pixel-based \cite{149} representation is generated for each image and each super-pixel is classified according to the preliminary GrabCut distributions. Across all of the images in a class or category, the aggregated sets of foreground and background super-pixels are used to train an SVM model to predict the relative probabilities for each super-pixel being classified as foreground or background. Thus, a unified model is learned to co-segment all images in the class or category. The final stage uses the SVM-predicted super-pixel probabilities as potentials for a single graph-cut segmentation on each image. A multi-class optimization that encourages a shared background model for the whole dataset with individual foreground models for each class is also explored, however, only a modest improvement in performance is observed as the shared background density leads to misclassification (e.g. sky or water can be mis-classified as foreground in images of one class due to the presence of blue in the foreground of another class). In a follow-on paper, Chai, et al. proposed TriCoS (Tri-level Co-Segmentation) \cite{36}, which aims at minimizing losses at three different levels: (i) the category level for foreground/background consistency across images belonging to the same category; (ii) the image level for spatial continuity within each image; and, (iii) the dataset level for discrimination between classes.
Chai [33] further proposed a model that unifies parts alignment and foreground segmentation into a convex framework.

Angelova, et al. [8] proposed a super-pixel based segmentation algorithm. A global segmentation model is trained per-domain using super-pixel regions. Features such as average color, HOG, and shape mask (normalized to a 6x6 region) are used. Once an initial set of foreground regions is predicted, the mask is grown with Laplacian propagation. Features from the masked segmentation region and from the whole image are extracted and concatenated. In [7], Angelova, et al. used a template-learning method similar to [204]. Multiple such templates, as well as globally-pooled features (HOG), are combined and weighted for optimal classification results. Angelova, et al. [6] further compared three popular directions for FGVC, namely segmentation, region pooling (part-based pooling), and direct optimization of the multi-class cost function. They found that these methods are complementary and enhance performance when used together.

NilsBack, et al. [137] applied bag-of-words features to a newly-created fine-grained flower dataset. Binary segments are extracted from all images using the contrast-dependent prior MRF (Markov Random Field) cost function of [21], and then optimized using graph cuts. Three separate vocabularies for color, shape and texture are constructed and combined as a unified representation. A nearest neighbor classifier is trained and the top ranked images are evaluated. Nilsback, et al. [138] introduced an additional algorithm for flower segmentation. An initial segmentation is implemented with a binary MRF using a contrast dependent prior. A flower shape model is fit with the assumption that all petals should agree on the flower “centre”. A concave corner finder is proposed which uses a “worm”, effectively a ratio of the line integral (distance) between two points along the contour to the straight-line distance between the two points. Center candidates are estimated as the intersection of two petal mid-lines, and consensus (low mid-line deviations for the other petals) is sought to predict the final center. Once the petals have been found a new segmentation model can be trained and applied. This process is repeated until convergence. Nilsback, et al. [139] and Nilsback,
et al. [140] are follow-up papers with various modifications and improvements to the original algorithms.

Sfar, et al. [158] sought to leverage aligned “vantage frames” for leaf image recognition. Vantage frames are determined by combinations of domain-specific landmarks, a method inspired by botanists. The object centroid is computed on the mask generated by the Ostu segmentation algorithm; the object scale is then taken as the radius of the bounding circle. The orientation of the vantage frame is the direction from the landmark points to the image centroid and the unit distance to be the approximate scale of the object. Once frames are found (and they are with very high accuracy), color, shape and texture parameters are extracted for each such frame. For fine-grained recognition, a taxonomy tree is built and for each tree node, a binary SVM is built to determine if each given image belongs to the node. A breadth-first search is performed and only positive nodes that produce high probabilities proceed to the next level. Finally, the fine-grained species corresponding to the positive leaf nodes are sorted according to likelihood ratio.

Parkhi, et al. [145] applied the detection approach proposed in their early work [144] to the fine-grained classification of pets (cats/dogs). The Oxford-IIIT PET dataset was created and used for evaluation. The basic strategy is to use a DPM (deformable part model [58]) trained on annotated head boxes to detect heads in the test set. The detected head is then used to seed an image-specific segmentation model and detect the rest of the cat/dog. A 4000-visual word vocabulary of SIFT descriptors is then used to quantify features in various pooling regions of different spatial histograms to achieve bag-of-words descriptions for these regions; finally, an SVM is used for classification.

Belhumeur, et al. [12] described an end-to-end vision system for leaf recognition. Leaf images are segmented from the background (the method requires a solid white background), and then matched to known shapes in a database. The segmentation uses EM (expectation maximization), initialized via k-means (k=2), on 2D (saturation-value) pixel values, to split pixels into two groups. K-means is initialized with the median of pixels near the boundary
for the background, and the median of the pixels near the center for the foreground cluster. All pixels are classified based on the two Gaussian densities learned by EM, and the largest foreground component is retained as the object. After segmentation, the IDSC (Inner-distance shape context) from Ling and Jacobs [117] is used for shape matching.

In follow-on work Kumar, et al. [102] formally described Leafsnap, a deployed vision system (iOS app), used for identification of trees by their leaves. The approach requires well-matted photographs with a solid light-colored background to assist the segmentation. The first step is a preliminary classification (SVM on gist features) to tell whether the image affords a good leaf silhouette, and if so, a segmentation is done to get the leaf mask. Curvature features are measured, basically using a disc on contour points to see the fraction of overlap in area/perimeter with the silhouette. These measures are taken over the entire contour with different sized disks. A histogram is computed of curvature over scale (HoCS) and this feature is used with nearest-neighbor matching for classification.

Krause, et al. [99] proposed the idea of “align by segmentation”. A GrabCut-based co-segmentation method is applied to segment foreground from background, utilizing bounding box information to refine the segmentation. A pose graph is constructed where nodes are images and edges indicate similarity between image poses. Alignment is done by propagating points along the graph that correspond to the same object locations. A diverse set of parts are selected by k-means cluster centers. In the recognition step, different weights are assigned to different features for discriminative learning using an SVM with sparsity constraints.

### 6.3.2 Human-In-the-Loop

Humans are usually good at certain tasks at which computer vision algorithms struggle, for instance, localizing object keypoints or recognizing semantic attributes like ”does this bird have blue eyes?”. Computers, on the other hand, excel at sifting through and learning from millions of images, classifying into hundreds or even thousands of categories, or performing object recognition at large-scale automatically. Human-in-the-loop approaches advocate a
Figure 6.3: Human in the loop [27] methods aim to tackle FGVC through the joint efforts of computer vision algorithms and human intelligence.

hybrid human-computer system that combines the respective advantages of computer vision algorithms and human expertise. Such systems can be practical intermediate solutions towards integrating contemporary computer vision algorithms for fine-grained visual categorization.

In one of the seminal FGVC papers, Branson, et al. [22] introduced an interactive fine-grained recognition framework similar to the visual 20 questions game. Human responses to low-level questions like “is the bird’s belly white?” are collected. To minimize the number of questions asked of human annotators, the most effective question should be selected at each point in time. A probability model is proposed to model the information gain of each candidate question. Using Bayesian inference under certain independence assumptions, the computer vision model and human responses are incorporated into the probability model. The authors gathered responses to 25 visual questions composing 288 visual attributes for the 200-category bird dataset they introduced. “Deterministic” class-attributes were collected from a website called what-bird.com. Subsequently, Branson, et al. [23] presented a framework for large-scale annotation and learning structured models. A DPM with HOG-based part detectors is proposed and dynamic programming is employed to solve the resulting maximum likelihood problem. The task is then formulated as a max-margin structured learning problem and the optimal solution is found by gradient descent. The maximum likelihood location of each part is displayed via an interactive user interface in real time. The user is then asked to fix any incorrectly predicted parts. The correct locations are then fed back into the model for parameter updating. As a result, the average number of parts needing manual
annotation reduces to 3.9 with a training dataset of 4000 images. Branson, et al. [25] also explored how one type of annotation (e.g. segmentation) benefits another (e.g. parts) with the help of computer vision algorithms and human annotators. As annotators label images, the proposed algorithm incrementally learns a translator from source to target labels as well as a computer-vision-based structured predictor. These two components are combined to form an improved prediction system which accelerates the annotators’ work through a smart GUI. The proposed tasks include labelling bounding boxes, segmentations, 2D and 3D parts, as well as class and attribute labels. Branson, et al. published the journal [27] version of [22, 182] with several extensions to the original work. In [28], Branson, et al. proposed an intelligent algorithm to jointly consider human labelers and computer vision algorithms for better annotation. This is done using a sequential estimation of risk over a probabilistic model that combines workers’ skill, image difficulty, and an incrementally trained computer vision model. Specialized models and algorithms are developed for binary attributes, part keypoints, and bounding box annotations. The proposed method can reduce annotation time by a factor of 4x - 11x for binary filtering of web search results, 2x - 4x for bounding box annotation of pedestrians in images, while in many cases also reducing annotation error.

Wah, et al. [182] focused on two tasks for human annotators: clicking on object parts and answering binary questions. Given the input and user responses, the possible class prediction is modeled as a posterior probability. By adding a new parameter for the keypoint location or binary attribute, computer vision algorithms (part detection and attribute detection) and user responses (part clicking and attribute question answering) are incorporated into the probability model. To select the next question exposed to human annotators, the information gain equation from [22] is adapted to consider the above two forms of responses. Additionally, minimizing overall user time is included as a constraint for question selection. Wah, et al. [184] proposed a second system that works with non-expert annotators, this one requiring no attributes (which are expensive to obtain). Given a query image and a set of candidate images, the annotator is asked to select the most similar candidate. A
semantic embedding space is learned to preserve the induced perceptual similarity. At test time, the learned computer vision algorithm and human inputs are combined to refine the final result.

Deng, et al. [45, 46] proposed a game called “Bubble Bank” that allows human annotators to gradually reveal annotator-chosen regions in a blurred image to felicitate completion of a two-category classification task. It was found that non-expert humans are able to make accurate classifications based on only a few parts and that the differences between classes are highly local. For automatic classification, human-collected bubbles are used as templates, and when convolved with a new image, the convolution response maps form a novel representation.

Cui, et al. [41] proposed a generic iterative framework for fine-grained categorization and dataset bootstrapping. Combining deep metric learning with humans-in-the-loop, a low dimensional feature embedding with anchor points is learned on manifolds for each category. These anchor points capture intra-class variances and remain discriminative between classes. In each of several rounds, images with high confidence scores are sent to humans for labeling. By comparing with exemplar images, labelers mark each candidate image as either a “true positive” or a “false positive.” True positives are added into the current dataset and false positives are regarded as “hard negatives” for the metric learning model. The model is then retrained with an expanded dataset and hard negatives for the next round.

6.3.3 Features and Pooling

Feature extraction and pooling are two of the most fundamental components in both traditional and modern computer vision algorithms. Before the deep learning era, BOW (bag-of-words) representations and their variants were dominant in image recognition. BOW, in its simplest form, contains several steps: local descriptors such as SIFT or HOG are extracted from interest points or regularly-spaced grid patches; a vector quantization step is performed to assign patch descriptors to a set of predetermined clusters (also called a vocabulary, dictionary...
or codebook); a histogram is constructed by counting the number of patches assigned to each cluster; and finally a classifier such as SVM or Naive Bayes is applied for the final classification. Fisher Vectors are a popular BOW variant that characterizes a signal with a gradient vector derived from a probability density function (pdf) which models the generation process of the signal.

To address the challenges brought by fine-grained visual categorization, traditional algorithms seek to learn more discriminative features. Examples along this line include fusion of multiple features like color and shape, better dictionary learning algorithms, etc. As modern algorithms automatically learn features via neural networks, approaches have shifted to the development of specialized pooling layers. One of the most influential works is bilinear pooling [115], where a special pooling layer is proposed to combine features from two underlying networks.

**Hand-crafted Feature Engineering**

Khan, *et al.* [94] proposed to fuse color and shape cues in a compact and discriminative way. Because an intuitive way to combine different features, the Cartesian or “outer” product, yields a very high-dimensional representation, the Divisive Information-Theoretic feature Clustering (DITC) algorithm is applied for dimensionality reduction. A weighting parameter $\alpha$ is introduced to control the relative contribution of color and shape cues. The classifier is a non-linear, multi-way, one-versus-all SVM using the $\chi^2$ kernel.

Chen, *et al.* [38] proposed the Generalized Hierarchical Matching framework for image classification, which extends pyramid matching and integrates side information like object confidence maps and visual saliency maps. Gao, *et al.* [161] proposed to learn a shared dictionary as well as category-specific dictionaries that encode category-specific features to capture useful discriminative features. Xie *et al.* [198] proposed a new local descriptor based on SIFT called RIDE (Reversal Invariant Descriptor Enhancement) that is invariant to horizontal image flipping. A descriptor transformation function is designed to remain
unchanged for reversed image patches. The orientation of each descriptor is computed as an approximated summation on the gradient-based histograms.

Random forests [129, 206] have also been used in fine-grained visual categorization. In Martinez-Munoz, et al. [129], a random forest is trained to predict the class of an image based on individual keypoint descriptors. To categorize a new image, descriptors for all detected keypoints are “dropped” through the trees, and the class evidence at each encountered leaf node is summed to obtain an overall evidence vector. This is then sent to a second-level classifier to make the categorization decision. Yao, et al. [206] proposed a random forest algorithm with discriminative decision trees for FGVC. For each node as a random decision tree is grown, a set of region descriptors is selected that splits the input images into two categories.

Senchez, et al. [156] proposed to employ Fisher Vectors (FVs) for FGVC because they contain high-order information and are both discriminative and scalable. Gosselin, et al. [75] described the winning entry in the FGComp 2013 competition based on the FV. The algorithm pipeline includes: resizing images to S total pixels (instead of fixed length on smaller dimension); extracting SIFT descriptors densely at multiple resolutions; post-processing the SIFT descriptors (PCA, RootSIFT, etc.); encoding with the FV; spatial pyramid pooling; and post processing with signed power-law normalization. The resulting vector is $l_2$-normalized and cosine similarity is used as the similarity metric. Last, a 1-vs-rest linear SVM classifier is trained and used for classification.

Deep Feature Pooling

Lin, et al. [113–115], the bilinear pooling approach mentioned earlier, proposed a novel network architecture that consists of two feature extracting trunks whose outputs are combined via outer product (cartesian product) to obtain “bilinear features”. Bilinear features capture pairwise correlations between the extracted features and can be used to model part-feature interactions, e.g. if one of the networks is a part detector and the other is a local feature
extractor. Sum-pooling is used to aggregate the bilinear features across the image, which is then passed through a signed square root layer and an $l_2$-normalization layer, which is shown to improve performance in practice. The major problem of bilinear pooling is its high memory consumption; the layer output dimension grows quadratically with extracted feature size. Gao, et al. [63] therefore proposed a compact bilinear pooling layer. Bilinear pooling, as a second-order polynomial kernel, is approximated by randomized compact features such that the inner product between two input samples remains the same after randomization. Random Maclaurin projection and Tensor Sketch are used and achieve the same performance with a smaller memory footprint. Cui, et al. [42] extended bilinear pooling and compact bilinear pooling to higher order kernels. Gaussian RBF kernels are shown to approximate up to a given order by Taylor series expansion with compact polynomial kernel feature vectors. The count Sketch algorithm (involving FFT and IFFT) is used for computing polynomial kernel approximation features. Simon, *et al.* [164, 165] proposed to combine bilinear pooling and sum pooling into a unified pooling layer, called alpha-pooling. A learnable parameter $\alpha$ and a unified representation are introduced, enabling a continuous shift between bilinear pooling and sum pooling. Yu, *et al.* [211] extended bilinear pooling to all of the intermediate layer feature maps. Kong, *et al.* [96] proposed to represent covariance features as a matrix and apply a low rank bilinear classifier for dimensionality reduction. Wei, *et al.* [192] proposed a bilinear pooling variant that transforms the CNN feature matrix to an orthonormal matrix.
consisting of its principal singular vectors, which lies on a Grassmann manifold. Cai, et al. [31] proposed to apply a polynomial kernel-based predictor to network activations and integrate multi-layer activations via kernel fusion. Engin, et al. [55] proposed a deep network that jointly learns local descriptors, a kernel-matrix-based SPD (symmetric positive-definite) representation, and the final classifier via an end-to-end training process.

Wang, et al. [189] proposed an asymmetric two-stream architecture. The P-stream contains a $1 \times 1$ convolutional layer followed by a GMP (global max pooling) layer to learn patch-level discriminative features. The G-stream is designed to capture global-level information. The two streams are combined in the end. A Cross-Channel Pooling layer followed by a softmax loss layer is applied to guide the P-stream in learning patch-level features. A non-random layer initialization strategy is applied to avoid degradation.

### 6.3.4 Parts and Part-Normalization

As the subtle differences between visually-similar categories often lie in highly-localized regions, discriminative features are better encoded by part features rather than a holistic object representation. Part-based representations decompose the object into a collection of its semantic parts; for example, a bird can be represented by parts such as beak, feet, wings and tail of a bird. Part locations are either provided via annotation or are learned. Algorithms to localize object parts include templates/exemplars, part detection and keypoint localization. Features are often extracted from both part-level and image-level patches and are concatenated to then be fed into a classifier.

### Parts / Part-Localization

Early work on part-based representations focused on hand-crafted features. Liu, et al. [119] proposed a dog breed recognition algorithm focusing on dog faces. An SVM regressor trained on gray-scale SIFT descriptors is used as the dog face detector. Different locations and scales are searched and non-maximal suppression is performed. The eyes and nose are found by the
consensus of an exemplar based detector and a geometry model. SIFT features from multiple object parts and a 32-dimensional color histogram are concatenated and fed into a one-vs-rest SVM for classification. Liu, et al. [118] also proposed an exemplar-based approach for bird part localization. The part locations are predicted by matching likely exemplars that satisfy globally-plausible configurations and are consistent with pose and subcategory appearance. A pose detector is constructed by grouping part configurations into tight clusters, and similarly, a subcategory-specific detector is constructed by grouping images in the same subcategory. Pose detectors are responsible for finding parts and subcategory detectors focus on verifying the hypotheses. Krause, et al. [97] proposed the concept of “ensemble of localized learned features (ELLF)”. In contrast to bag-of-words approaches, ELLF compares the appearances of each part, aggregating similarities together via concatenation. Part discovery is performed by: (i) choosing a seed image at random and retrieving its nearest neighbors (in terms of HOG features); (ii) parts with the highest energy (measured by HOG variance across images) are selected; (iii) a latent SVM detector is learned for each selected part; (iii) an ensemble
of parts is obtained by repeating this process. At test time, all of the detectors are used; they either fire – and appearance features are extracted with a CNN – or are zeroed out if not visible. A linear-svm is used for final classification. Zhang, et al. [220] proposed to detect the less deformable parts using a template-based model and to infer other parts by geometric alignment of the foreground mask under a semantic prior. Similar subcategories are fused iteratively using a neighbor joining method. Vedaldi, et al. [181] introduced the FGVC-Aircraft dataset with both part segmentation and visual attribute annotations. A hierarchy of filters is learned to reduce the number of filter operations used for part detection. A coarse-to-fine algorithm is developed that begins with rough score estimates for sets of similar filter components, and then recursively refines such scores by working with increasingly smaller sets of filter components. Chai, et al. [37] proposed to symbiotically combine segmentation and part-localization – by minimizing a joint energy function – improving both, as if by a feedback loop between them. The approach is implemented with a DPM to find part regions and couples a segmentation/saliency map for each DPM part (the root and all parts).

Other works employed deep neural networks for part/keypoint detection, feature extraction and classification. These steps can be implemented by multiple separate networks, or, they can be fused into an end-to-end learnable model. Zhang, et al. [218] applied the R-CNN model to train both object and part detectors from bottom-up region proposals. During test time, all windows are scored by all detectors, and non-parametric geometric constraints are applied to rescore the windows and choose the best object and part detections. Huang, et al. [85] employed a fully convolutional network to detect object parts as 2D keypoint locations. A two-stream CNN architecture is proposed: the part stream is composed of several CNNs, independently trained on object parts; the object stream utilizes bounding-box level supervision to capture object-level features. Xie, et al. [197] proposed a hierarchical structure learning algorithm to find mid-level concepts beyond basic parts. A geometric phrase pooling algorithm is then used to capture mid-level structures in the local feature groups. Zhang, et al. [213] proposed a unified model for semantic part detection, feature
extraction and classification. Several sub-networks are combined into one hybrid network with K-NN used for part proposal and Fast R-CNN used for detection. A semantic part ROI pooling layer is used to pool the features corresponding to a particular part. Shih, et al. [162] replaced the final linear layer with two separate layers for keypoint localization and visibility respectively. The final loss is the sum of the localization and visibility losses. Keypoint predictions from different image crops are aggregated; low visibility predictions are discarded and the resulting medoid is used as a robust keypoint estimator.

Pose-Normalization

Compared to traditional part-based approaches, pose-normalized representations aim for complete invariance to object pose and camera orientation. The extracted representation of a given object or category will ideally be consistent across any range of images, in spite of each image having its own object pose and camera viewing angle. In the context of fine-grained recognition, pose-normalization can be viewed as effectively factorizing the visual representation of a category into a domain-level geometric model (coarse shape, configuration, kinematics) and a subcategory-level model of appearance (color, texture, fine shape variation).

An object’s pose – the configuration of its parts – is estimated and these geometric predications define a mapping from the observed image onto a pose-normalized version of the object’s surface, thus extracting a consistent, pose-independent representation of the object’s appearance. Very few approaches explicitly model the object’s geometry; one such approach is Birdlets, proposed by Farrell, et al. [57]. Some methods use more implicit pose-normalization, often predicting part locations and then extracting local part patches at the respective part locations. Pose-estimation is typically easier with an implicit representation, and therefore generally results in a more accurate final classification.

Farrell, et al. [57] proposed a method based on Poselets [19], but with 3D parts (volumetric prolate ellipsoids for the head and body) to train HOG-feature “Birdlet” detectors. Part configuration is guaranteed to be consistent by the estimated ellipsoid parameters
(locations/orientation/scale). Once the ellipsoid parameters are estimated, features are extracted as though normal to each ellipsoid (projected onto its surface) and a random-forest (RF) based classifier (one per part) is used; the actual features in the RF are formed by concatenating location on the ellipsoid with the SIFT descriptor of the surface-normalized region.

Berg, et al. [13] proposed POOF (part-based one-vs-one features) where a region defined relative to two keypoints is cropped and transformed to classify between two categories. The region is tiled with a grid of cells. Low-level image features (HOG) are extracted and a linear SVM classifier is trained. Grid cells are thresholded by relative importance and a connected-component region is found. Liu, et al. [120] proposed a part-pair based representation for FGVC, similar to POOF, except that the part-pairs are not known, but are detected. A bottom-up method is applied to roughly locate the part location using a cascade of weak classifiers. A meta part detector is built to use the collective information from all part pairs. Built on part detection results, rigid and flexible part configurations are applied to further filter out noise and get accurate localizations. Pang, et al. [143] proposed to extract image features similar to POOF. A small group of hand-chosen part pairs are selected to reduce complexity. Sparse coding is used instead of a lossy codebook-based method. A discriminative model is learned to assign different weights to the different patch features, and a linear one-vs-all SVM is learned as the final classifier. Guo and Farrell [76] proposed a deep network pipeline to extract pose-aligned features like POOF. A fully convolutional network is trained to predict object keypoints. Features from rectangular patches, anchored by two keypoints, are extracted and fed into a tiny MLP network for final classification.

Zhang, et al. [216] performed beam search to prune a large set of poselets down to \(100\). BOW-SIFT and PHOW (pyramidal histogram of words) are used to extract features from the pruned poselet activation patches. To model the similarity between features, warping kernels are learned and poselets are clustered. Zhang, et al. [217] proposed two deformable part descriptors (DPD): the DPD-strong leverages the semantics inherent in
strongly-supervised DPM (deformable part model) parts; the DPD-weak exploits semantic annotations to learn cross-component correspondences, computing pose-normalized descriptors from weakly-supervised DPM parts.

Branson, *et al.* [26] applied pose-normalization to CNN features. Candidate parts are seeded with one keypoint and expanded to a bounding box that includes the $M$ nearest other keypoints. A minimal set of prototypes is picked such that all keypoints are effectively covered. Pose estimation is done with a DPM-based method (see [24]) which produces estimates of the keypoint locations. CNN features are then used to describe the prototype pooling regions and an SVM is used on the concatenated features for classification.

In Huang, *et al.* [84] a set of polygons are generated that are composed of multiple parts. For each polygon, a classifier is trained based on deep features of a convolutional network. A greedy algorithm is then employed to select the discriminative and complementary polygon-based classifiers that deliver the highest classification accuracy for fine-grained object categories.

Lin, *et al.* [112] proposed a unified model called Deep-LAC with three sub-networks: a part localization sub-network, an alignment sub-network and a classification sub-network. The part-localization sub-network regresses part bounding box coordinates given ground truth locations. The alignment sub-network receives part locations from the localization module, performs template alignment and then feeds a pose-aligned part image for classification (a single linear layer). The proposed Valve Linkage Function defines the output of the alignment sub-network, linking the localization and classification modules and back-propagating the classification error to both modules.

Tang, *et al.* [175] revisited pose normalization for few-shot fine-grained recognition, showing that it: (i) improves accuracy by 10 – 20% for both shallow and deep architectures; (ii) generalizes better to new domains; and, (iii) is effective across multiple few-shot algorithms and network backbones.
Unsupervised Templates / Part Exemplars

One of the limitations of part-based and pose-normalized representations is that they require manual annotation of object parts. These annotations can be expensive to obtain or even hard to clearly define. Many efforts have been made to overcome such limits by discovering object parts in a weakly-supervised or unsupervised manner.

Yang, et al. [204] proposed to extract random patches/templates from all training images (100 templates for each of 420 training images) and to use them as a basis for representing all images. Color and gradient features were used; different SVM classifiers are trained on disjoint subsets of the feature dimensions and the results of these classifiers are simply averaged to produce a final prediction on each test image.

Lee, et al. [108] aimed to find mid-level visual detectors that contain vital information and are insensitive to style changes. Random patches are cropped and HOG features are extracted; clusters are then formed by unsupervised grouping. Additionally, an SVM classifier is trained to include more positive samples from images of the same label to better account for style change. The outcome of SVM scores are concatenated and fed into an SVM regressor in order to get the final classification accuracy.

Gavves, et al. [64, 65] proposed two methods for alignment, one supervised, the other unsupervised. In both cases, a grabcut is initialized on the bounding box (required annotation) to get an object mask. For supervised learning, the HOG features of the mask boundary are computed and the nearest neighbors from the training set are found. In this manner, annotations (ground truth part locations) from a training image are transferred to a test image and features are extracted from windows enclosing those part locations. For unsupervised learning, an ellipse is fit to the silhouette and, with a gravity vector assumption, a coordinate frame is fit to the object mask. Relative to this coordinate frame, pooling regions are computed (bands along the principal vector) and features are extracted. Fisher Vectors are used as features in both settings; the unsupervised approach turns out to be more accurate.
Goring, et al. [74] proposed to perform part localization by matching a query image to similar training images and transferring the training image annotations to the query image. This is done, in practice, by cropping all images to their bounding boxes, resizing to 256x256 and then computing HOG features. This HOG-based description is used for lookup in the training set and the part locations from the $K$ nearest training images are then transferred and the appearance is extracted from the locations in the query image. Appearance is characterized by shape (opponentSIFT) and color (map of “color names”) and classification is done after pooling the histograms via a $\chi^2$ kernel. Global features are also included relative to the whole bounding box. Freytag, et al. [59, 60] proposed a method very similar to [74]. Local Learning creates a model on the fly that is tailored to each given test image. This method finds seed patches from a predicted segmentation and uses them to learn patch detectors. These initial patch detectors are then used to find additional training patches via bootstrapping with convolutions. A query- or “exemplar”-specific representation is then generated; query patch detectors are used on the “neighboring” training images to get responses for each detector and a vector of these responses is created. A linear SVM is trained on the neighboring training images and the query image is classified.

Wang, et al. [188] proposed to discover discriminative triplets of parts (patches). The proposed triplets consist of three appearance descriptors and two representative geometric constraints. Candidate triplets are initialized from sets of neighboring images and selected by how discriminative they are across the training set. The mid-level representation consists of the maximum responses of the selected triplets with geometric constraints, which is fed into a linear SVM for classification.

In Yao, et al. [208], the less deformable part (such as the head of a bird) and the object itself are first detected using template-based detectors. The confidence map is obtained by weighting the top scored detection locations of the object and then used as segmentation prior. Based on the localization and segmentation results, part alignment is performed to obtain several semantically meaningful parts, such as the head, body, and tail. Mid-level
features are learned for classification according to the part-based low-level representation. Similar subcategories are fused into macro-classes, and one-vs-all SVMs are learned.

Main-stream weakly-supervised part discovery methods make use of the intermediate-layer representations in discriminatively-trained neural networks, which carry rich part-level semantics and can be seen as noisy part detectors.

Simon, et al. [163] proposed an unsupervised algorithm for part detection. Based on the observation that intermediate-level feature maps are a good indicator of the existence of certain parts, a part proposal is computed by back propagated gradients from the max-value location of certain feature maps to the image space. Given all the proposals, a joint model is constructed to minimize the deformable part model similar energy function. Zhang, et al. [222, 223] proposed to select part detectors from the trained CNN model, looking for deep filters with significant and consistent responses are selected. Positive samples are chosen according to filter activations and a set of discriminative detectors is trained iteratively. Zhang, et al. [225], proposed a weakly-supervised part-based representation, where a multi-max pooling strategy is devised to generate features for multi-scale part proposals by leveraging the internal structure of a CNN. All part proposals are clustered and useful clusters are selected and encoded into a global image representation using scale pyramid matching.

Ge, et al. [66] proposed to build complementary part models in a weakly-supervised manner for fine-grained recognition. Given image-level labels, rough object instances are extracted by weakly-supervised object detection and instance segmentation using Mask R-CNN and CRF-based segmentation. The best parts for each object instance are estimated by preserving as much diversity as possible. A bi-directional long short-term memory (LSTM) network is built to fuse and encode the partial information of these complementary parts into a comprehensive feature for image classification.

Yao, et al. [210] proposed an algorithm to extract part-level and object-level features with only category labels. For part-level features, selective search is applied to generate
candidate bounding boxes and an objectness map is generated using a CNN. The most distinctive local regions are selected from the bounding boxes with the help of the objectness map. For object-level features, a graph analysis algorithm is proposed to co-localize the object using images containing similar objects.

Liang, *et al.* [111] proposed a Gaussian mixture model that fuses part features with a Gaussian mixture layer. Part proposals generated by selective search and feature maps from middle layers are used for calculating part features via spatial pyramid pooling. The Gaussian mixture layer finds clusters for the input and generates output features based on a combination of cluster centers that is used for classification.

Huang and Li [86] proposed a method consisting of part segmentation, region feature extraction and attribution, and attention based classification. A U-shaped prior distribution is enforced for the occurrence of each part. This is done by minimizing the Earth Mover’s Distance between the prior and the empirical distribution of part occurrence.

Jaderberg, *et al.* [87] introduced the spatial transformer network that learns a set of transformation parameters which, when applied on the feature maps, makes the model invariant to translation, scale, rotation and more generic warping. The spatial transformer layer consists of three components: (i) the localization network that takes the input feature map and outputs the transformation parameters; (ii) a sampling grid that is computed given the parameters; (iii) a sampler that outputs the sampled feature map based on the sampling grid.

Kanazawa, *et al.* [93] presented WarpNet for weakly-supervised pose alignment and single view 3D reconstruction for fine-grained datasets. The network takes two images as input, where the target image is a warp of the source image with Thin Plate Spline (TPS) transformation, whose parameters are predefined. The output of the network is a deformed grid that parametrizes a TPS transformation, a step similar to a spatial transformer network. The generated TPS is then used to warp a point in the source image to the corresponding point in the target image. Their $l_2$ distance is minimized to provide supervision for parameter
6.3.5 Attention

Humans are able to focus their most powerful visual processing resources onto a selected part of the visual field [62], and build a comprehensive representation of the physical world over time [150]. Attention mechanisms in computer vision [11, 132] work by sequentially “attending” to a series of image patches in search of discriminative information, making it a natural fit for Fine-grained Visual Categorization (FGVC).

Sermanet, et al. [157] was perhaps the first to apply attention mechanism to FGVC, building on top of [11]. The proposed model is a recurrent network with $N$ steps; each step takes a coordinate $l_i$ and a composite image made of multi-resolution image patches centered at $l_i$. The network either predicts the next attention coordinate $l_{i+1}$ or produces the classification score. A Glimps network, composed of GoogleNet, is used for visual feature extraction and a fully connected network is used for location feature extraction. The visual and location features are then combined by concatenation instead of element-wise multiplication. A “vanilla” RNN is used instead of an LSTM and is followed by an Emission network for the next coordinate prediction. A Context network is used to provide an initial state.
Xiao, et al. [195] proposed a two-level attention model that extracts both object- and part-level features. A FilterNet is proposed to identify objects in a basic-level category. It uses selective search [218] for object proposal and the results are fed into the part-level detectors. A DomainNet is proposed to use mid-level CNN filters (Alexnet Conv4-layer) as part detectors. The selected patches then go through a part-level classifier which, together with the object-level classifier, provides the final classification.

Fu, et al. [61] proposed the recurrent attention convolutional neural network (RA-CNN) for fine-grained recognition. RA-CNN is a stacked network whose inputs range from full images to smaller regions at multiple scales. These multi-scaled networks share the same network architecture but have distinct parameters. Each network consists of (i) a classification sub-network and (ii) an attention proposal sub-network, used for feature extraction and predicting the next attentional location. The finer-scale network then takes an amplified attended region as input. The recurrent network is alternately optimized by softmax loss for classification and pairwise ranking loss for the attention proposal sub-network. The ranking loss forces the finer-level networks to attend better on discriminative regions and thus produces higher probability for the ground-truth class.

Zheng, et al. [229] proposed the multi-attention convolutional neural network (MA-CNN) with convolution, channel grouping and part classification sub-networks. The channel grouping network takes feature maps as input, and generates multiple parts by clustering, weighting and pooling the activations of spatially-correlated channels. The part classification network refines classification estimates for each individual part via spatial pooling. Two losses are proposed: (i) a classification loss on each generated part and (ii) a grouping loss to encourage the generated part heatmaps to be compact and diverse.

Zhao, et al. [227] argued that attention-based methods are limited by their inability to attend to multiple regions simultaneously and highly-localized regions. A diversified visual attention network (DVAN) is proposed. DVAN first localizes several regions of the input image, at different scales, and takes them as the “canvas” for the following visual
attention step. A convolutional neural network (VGG-16 in their case) is then adopted to learn convolutional features from each canvas region of attention. A diversified visual attention component is introduced to predict the attention maps, and a diversity loss is introduced to help attend to multiple regions.

Sun, et al. [169] proposed to learn multiple features for the attention region through a one-squeeze multi-excitation (OSME) module, which is applied to the last residual block’s input. The squeeze phase uses global average pooling to produce a vector whose size equals the channel number. In the multi-excitation step, a gating mechanism – consisting of a linear operation, sigmoid and ReLU – is independently employed. The output is used to reweight the input. A multi-attention multi-class constraint is employed to guide the attention regions toward better prediction of the correct class labels.

Rodriguez, et al. [151] proposed a multi-layer/multi-scale attention approach where attention modules are paired with traditional convolutional blocks. An attention module takes an output from a convolutional block and applies three types of submodules – an attention layer, an output layer and a confidence gating layer – to it. The attention layer contains parallel attention heads to produce attentional masks, which are used to modulate the likelihoods aggregating them spatially into a single class likelihood vector per attention head. The final output of an attention module is obtained by a weighted average of the output probability vectors.

Yang, et al. [205] proposed a Navigator-Teacher-Scrutinizer Network, or NTS-Net; the Navigator is trained to select informative regions; the Teacher is used to estimate a confidence score (likelihood that the region belongs to the ground-truth class) and the Scrutinizer takes the best regions determined by the Navigator, together with the full image, and computes the class-probability distribution. The Navigator and Teacher are trained together such that the region informativeness scores and the confidence scores are ordered identically (the highest scoring informative regions also have the highest confidence among all candidate regions for
belonging to the correct class). The top-K regions are used for classification – the Scrutinizer network is trained via cross-entropy.

Lam [105] proposed an attention model called HSNet. An H (heuristic) layer is used to measure the informativeness of a series of bounding boxes and an S (successor) layer is used to generate the next batch of candidate bounding boxes. The two functions are unified via an LSTM network. The model can adapt to different levels of supervision, with or without part annotations.

Liu, et al. [122] introduced an attribute-guided attention localization scheme where the attention region localizers are learned under the guidance of part attribute descriptions. A reward strategy is designed that incentivizes learning to localize regions that are both spatially- and semantically-distinctive using reinforcement learning. A fully-convolutional attention localization network is learned for each part.

He, et al. [78] proposed a weakly-supervised learning algorithm for object localization in fine-grained domains that is done with an N-pathway end-to-end discriminative localization network and multi-level attention.

Luo, et al. [123] proposed Cross-X learning. Attention region features are generated via multiple excitation modules and a cross-category cross-semantic regularizer is proposed to regulate the feature learning by maximizing the correlation of attention features extracted by the same excitation module while decorrelating those extracted by different excitation modules. A cross-layer regularizer is also introduced to match the prediction distributions of mid and high-level features by minimizing their KL-divergence.

Zhang, et al. [214] proposed a framework consisting of several experts and a gating network; each expert takes as input the attentional region of the previous expert. Expert diversity is promoted by maximizing the KL-divergence of the normalized non-maximal parts of the predicted probability vectors.

Zheng, et al. [230] proposed a Trilinear Attention Sampling Network (TASN) under the teacher-student training scheme. A trilinear attention module takes feature maps as input and
generates attention maps by self-trilinear product. The attention-based sampler highlights the attended parts with high resolution. For each iteration, the attention-based sampler generates a detail-preserved image by a random attention map and a structure-preserved image by the averaged attention map. A part-net and a master-net are formulated as teacher and student respectively.

Ding, et al. [48] proposed a framework called the Selective Sparse Sampling network (S3N). S3N collects local maximums from class response maps in order to estimate informative receptive fields and learn a set of sparse attention regions. The underlying image regions are re-sampled to highlight both discriminative and complementary features.

Wang, et al. [186] considered FGVC on edge cameras, which transmit mixed acuity images to a server for the sake of power conservation. The proposed DRIFT model consists of three networks: (i) a backbone network for feature extraction; (ii) a foveation actor network to generate a sequence of fixation actions; and (iii) a classification network for image label prediction. The Deep Deterministic Policy Gradient by Conditional Critic with Coaching (DDPGC3) algorithm is proposed to guide the training procedure.

6.3.6 Hierarchy/Taxonomy

Categories in the natural world are not arbitrary but are organized in a hierarchical manner. One way to organize object categories is through the concepts of superordinate, basic-level and subordinate categories. Category hierarchy carries rich semantic meaning that can be utilized in developing algorithms for fine-grained visual categorization.

BarHillel, et al. [80] introduced the idea (later reiterated in [57]) that a set of parts defines/distinguishes basic-level categories, whereas the properties of these parts define/distinguish the subordinate categories. This paper aims to take a two-stage approach representing the basic-level category by the set of parts and their configuration, and extracting feature vectors for each part (effectively part descriptors) for classification.
Berg, et al. [15] introduced the Birdsnap dataset. The dataset has 500 species with between 69-100 images per species; it includes bounding boxes and annotations for 17 parts/keypoints. One-vs-most classifiers with POOF features [13] are proposed where the K most similar species are excluded from the negative set so that similar species are not ruled out as easily. Two visually similar classes can be distinguished by the most discriminative part-pairs. Berg, et al. [14] additionally proposed the task of predicting visually similar classes, finding discriminative features and parts between similar classes, ultimately generating a visual similarity taxonomy. POOF features are extracted and LDA is applied for dimensionality reduction while retaining class-discriminative information; the negative $l_1$ distance is computed as a class similarity measurement. Once key differentiating features for a pair of classes are determined, they are visualized as a pair of illustrative images, one from each species, with the region of interest indicated on both images.

Ge, et al. [67] proposed a system which groups classes into visually similar subsets and learns domain-specific features for each subset. Progressive transfer learning is applied to learn a domain-generic convolutional feature extractor from a large-scale dataset of the
same domain as the target dataset (fine-tuning); clustering is done with K-means on the conv5 and fc6 layer (after LDA).

Pu [147] formulated fine-grained recognition as a multi-task learning problem. Rather than trying to learn independent classifiers for all categories/labels, the proposed framework exploits inherent similarities between classes and learns them together.

Wang, et al. [185] observed that a subordinate-level label carries with an implied hierarchy of labels. The region of interests for different hierarchical labels are different. The saliency region of an image is extracted using the stacked final (VGG) conv-layer feature maps.

Goo, et al. [72] employed class hierarchy for improved classification accuracy. The last conv-layer of a CNN is modified to learn a per-class feature map. A super-class is learned by min-pooling operation across subcategory feature maps. Difference-pooling is applied between the response maps of the parent and its child subcategories.

Wu, et al. [194] proposed the concept of fine-grained label and that any given label can be further classified into some sub-classes as fine-grained labels, the samples of which have less intra-class confusion and bigger inter-class variance. The fine-grained labels are obtained through unsupervised means without any domain knowledge or annotations.

Taherkhani, et al. [171] addressed fine-grained image classification in a weakly-supervised fashion, whereby a subset of images is tagged by fine labels (i.e., fine images), while the remaining images are tagged by coarse labels (i.e., coarse images). A CNN with a supervised low rank self-expressive layer and a supervised sparse self-expressive layer is proposed to capture both global and local features.

Du, et al. [50] proposed a progressive training strategy that adds new layers at each training step to exploit information based on the smaller granularity information found at the last step and the previous stage. A simple jigsaw puzzle generator is proposed to form images containing information from different granularity levels.
6.3.7 Misc

Applications

Fine-grained visual categorization is popular not only because of its challenging nature (for academic research), but also its close relation to real world applications. This section covers work that applies algorithms to real scenarios.

Wegner, et al. [191] presented a vision-based system to detect and classify publicly observable trees. Two tasks were proposed: det2geo (Fast-RCNN is used to detect trees from sources like Google street view and aerial images, multi-view scores are combined into final detections) and geo2cat (cropped versions of each object are obtained at different zoom levels; each cropped region is then fed through a CNN feature extractor). Branson, et al. [29] extended [191] and presented a fully-automated tree detection and species recognition pipeline that can process thousands of trees within a few hours using aerial and street view images, publicly available from Google Maps.

Gebru, et al. [70] proposed a novel task to utilize google street view images for demographic/census estimation. Information about cars in a local neighborhood are gathered through car detection, car classification, and attribute prediction, which are used to predict household income, per capita carbon emission, crime rates and other city attributes. For example, a region’s income level can be inferred from the price of its cars.

Zhi, et al. [231] leveraged multispectral imaging for fine-grained powder recognition. A broad range of spectral wavelengths can discriminate powders with reasonable accuracy. The huge number of spectral bands causes problems for acquisition, storage and computation. A greedy band selection approach using nearest neighbor cross-validation as the optimization criterion is proposed. It significantly reduces acquisition time and improves recognition accuracy.
Research directions for FGVC have expanded beyond 2D images into 3D models. This section covers work leveraging 3D car models to learn a better representation for FGVC.

Mottaghi, *et al.* [134] proposed to perform object detection, 3D pose estimation, and FGVC jointly using a hybrid random field model, as solving one task is beneficial to the others. A coarse-to-fine hierarchical representation is proposed, where each level of the hierarchy represents objects at a different level of granularity. Object labels are represented by binary random variables, and the coarse viewpoint is denoted by a discrete variable denoting azimuth ranges. The continuous viewpoint space is decomposed into azimuth, elevation, distance, and occlusion. All these variables are assigned with potential functions and are inferred jointly at test time.

Lin, *et al.* [116] fit a domain-general 3D model to an image according to regressed locations of 64 landmarks. The 3D representation is based on an Active Shape Model, where PCA is used to determine the mean model and eigenvectors describing the variation. A Deformable Part Model (DPM) detector is used to seed the landmark regression. A modified Jacobian system is used to optimize the fit of the 3D model to the landmark predictions, and the model is refined iteratively while transitioning toward a class-specific model, not the domain-general one. The image/object is described in terms of feature descriptors which are then concatenated into a high-dimensional vector. HOG and Fisher vectors are considered as descriptors and a linear SVM is used for classification.

Sochor, *et al.* [166] collected a dataset called BoxCars from a group of surveillance cameras. 3D bounding boxes can be automatically obtained for each image. The normalization is done by unpacking the image into a plane. The viewpoint is encoded as three 2D vectors, together with a rasterized 3D bounding box, is passed into the neural network for classification.
Web Supervision/Additional Data

Due to the difficulty in data collection and annotation, fine-grained datasets tend to be moderate in size. Several efforts have been made to exploit additional outside data, either downloaded from the Internet or from another labelled dataset, for better performance. Images from the Internet are usually recognized as weakly-supervised data as they contain weak and noisy labels. Active learning can be used to expand the original dataset using model predictions.

Xie, et al. [199] augmented the target dataset with images from a search engine, which are called hyper-class images as they either belong to the super-class (e.g. “dogs” and “cats”) or share a similar appearance with the query image. A multi-task regularized learning framework is proposed to exploit the regularization between the fine-grained and hyper-class classifiers.

Xu, et al. [201] collected images from websites (Flickr) with noisy labels. A part-detection network is trained on small supervised datasets like CUB and then applied to the larger weakly-supervised web data. The detected part patches are used as additional training data to produce more powerful feature representations via fine-tuning. A multi-instance formulation is proposed to jointly train the final classifiers on both supervised and weakly-supervised datasets.

Krause, et al. [100] proposed to leverage free, noisy data from the web using simple, generic methods of recognition. A hybrid approach of web search, active learning, and human annotations is used to collect a huge dataset.

Transfer Learning

Transfer learning is a learning scheme where a model is pretrained on a source dataset and then fine-tuned a different target dataset. Features learned from a generic dataset like ImageNet, when adapted to other tasks by fine-tuning the weights, can dramatically improve the downstream task performance. Transfer learning has achieved astonishing results.
in fine-grained visual categorization by transferring the knowledge from source datasets. Domain shift has been an important issue for transfer learning which refers to the distribution discrepancy between the target and source datasets. Several works address this problem.

Gebru, et al. [69] proposed to address domain adaptation using multi-task learning with classification loss, attribute loss, and consistency loss.

Azizpour, et al. [10] investigated the influential factors of transfer learning by examining considerations such as network width, depth, early stopping, fine-tuning, etc. The distance between the source and target dataset is an important factor impacting transfer learning success. By optimizing the above mentioned factors, this paper attains better results on the CUB and Flowers datasets than the best non-convolutional methods.

Cui, et al. [43] reported the first place winning entry in the iNaturalist 2017 large-scale species classification challenge. Central to their success is a training scheme that uses higher image resolution and deals with the long-tailed distribution of training data with a second fine-tuning step on a subset containing more balanced data and using a small learning rate. A measure is proposed to estimate domain similarity via the Earth Mover’s Distance. Transfer learning benefits under this measure by pre-training on a source domain that is similar to the target domain.

Zhang, et al. [224] proposed MetaFGNet, trained using a regularized meta-learning objective. A sample selection scheme is designed to select images that are, semantically, more closely related to the target task.

Xie, et al. [196] proposed to leverage GAN-generated images to improve fine-grained recognition. A pre-trained Yolo v2 object detection model is used to detect coarse-grained objects in the original dataset. The cropped images are fed to the GAN generator to produce artificial data and a uniform distributed label is assigned to the generated images. The real and generated images are mixed and fed to a baseline CNN classifier and a feature-fused CNN classifier.
Self-supervised Learning

One important direction in modern FGVC is self-supervised learning, a learning scheme that sits between unsupervised and supervised learning. Like unsupervised learning, self-supervised learning does not rely on additional annotations; like supervised learning, however, self-supervised learning usually produces one or more loss signals which the model can use to update its parameters.

Sudowe, et al. [168] argued that the reliance on ImageNet pre-training creates a barrier for testing out new architectures. Self-supervised learning is proposed instead to initialize model weights and shown to achieve comparable results with ImageNet-initialized weights. Specifically, the PatchTask is proposed – given a square patch of pixels from the input, its origin is predicted out of $K$ possible locations, which are discrete positions within the input image, similar to the Jigsaw-puzzle solving task.

Chen, et al. [39] proposed a “destruction and construction learning” method for fine-grained recognition. Besides the usual classification backbone, another “destruction and construction” stream is added. The input image is split into local regions and shuffled by a region confusion mechanism. An adversarial loss is introduced to distinguish the shuffled images from the originals. A region alignment network is trained to recover the spatial relationship between local regions.

Ensembles

Algorithms that train and combine the results of multiple models for a final decision are called ensembles. The diversity of models is guaranteed by either different model architectures or training on different data splits.

Ge, et al. [68] trained $K$ expert DCNNs by partitioning the data into $K$ non-overlapping sets. The classification decision from each expert is weighted proportionally to the confidence of its decision. This defines a single network (MixDCNN), composed of $K$ sub-networks (expert DCNNs), that can be trained to perform classification.
Moghimi, et al. [133] proposed an algorithm to ensemble different models. The algorithm works by assigning different weights to different model outputs. The authors use GD-MCBoost for this purpose. In this framework, weak learners are separate CNN models that are trained to minimize a squared error loss.

Zhang, et al. [212] proposed to learn encoding parameters and codebooks through low-rank sparse coding (LRSC) with both general and class-specific codebook generation.

**Other**

Hendricks, et al. [79] endeavor to generate explanations for fine-grained recognition, explanations that should both reflect the image content and also contain discriminative information. The first part is ensured by relevance loss, similar to image captioning. The second part is ensured by a discriminative loss based on reinforcement learning.

Zhang, et al. [215], represented an image by a graph, called a graphlet, whose nodes are super pixels and whose edges indicate their spatial connectivity. To cluster sub-categories into super-categories, SIFT features are matched to compute pairwise similarity. A dense graph mining algorithm is proposed to discover graphlets representative of different categories (both sub-categories and super-categories). A hierarchical graphlet kernel is proposed that quantizes the extracted graphlets from an image into a 1D vector. An SVM is used for classification.

Dubey, et al. [51, 52] proposed to regularize deep neural networks by injecting pairwise confusion to address the inter-class similarity problem. A Siamese network is trained with a loss function that attempts to bring class probability distributions for images of the same class closer to each other.

Aoha, et al. [142] proposed to jointly model object categories, their spatio-temporal distributions, and photographer biases from presence-only observation data. The proposed loss contains three items that encourages: (i) the model to predict the presence of objects where they have been observed in the training set and downweight their likelihood where
they have not; (ii) the affinity between a photographer $p$ and a location $x$ be high if $p$ was present at $x$; and that (iii) a photographer has a low affinity for a category unless they have previously observed it.

Liang, et al. [110] proposed to prepare images batched as triplets and use a multi-task training approach with adaptive triplet and classification losses. A second-order distance function is used to capture local pairwise interactions of the embeddings. Luo, et al. [123] proposed two components: a cross-category cross-semantic layer (C3L) regularizer and a cross-layer (CL) regularizer. C3L applies global average pooling and $l_2$ norm to the input and computes a correlation matrix. The goal is to maximize the main diagonal values and suppress others to encourage the consistency of a given filter. The CL regularizer is proposed to match the prediction distribution between different layers. Ji, et al. [89] proposed to apply adaptive neural trees, a combination of neural network and decision trees, for fine-grained recognition. An attention transformer is used to learn discriminative features. Wang, et al. [190] tackled the problem of discriminative region diffusion in high-level feature maps. An end-to-end Discriminative Feature-oriented Gaussian Mixture Model is proposed with low-rank representation and representation reorganization mechanisms.

### 6.4 Datasets and Performance Evaluation

An important element of this comprehensive study is to survey, not only the diversity of proposed approaches, but also their relative performance. In order to objectively compare the performance of different approaches, it is crucial that all methods provide results on the same benchmark(s)/dataset(s). While a variety of fine-grained datasets have been introduced, very few have been widely adopted as standard datasets for comparing different approaches.

The Caltech-UCSD Birds (CUB-200) dataset, which features 200 different species/categories, is indisputably the most widely used for comparison of fine-grained recognition techniques. While an initial version of the dataset was used in Branson, et al. [22], the subsequent 2011 version [183] has emerged as the standard for FGVC. As evidence of this widespread adoption,
Table 6.1: Fine-Grained Visual Categorization (FGVC) Datasets

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Year Introduced</th>
<th>Number of Categories</th>
<th>Image Count</th>
<th>Number of Papers Reporting Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min</td>
<td>Median</td>
</tr>
<tr>
<td>CUB 2011 [183]</td>
<td>2011</td>
<td>200</td>
<td>41</td>
<td>60</td>
</tr>
<tr>
<td>Stanford Cars [98]</td>
<td>2013</td>
<td>196</td>
<td>48</td>
<td>84</td>
</tr>
<tr>
<td>FGVC Aircraft [124, 181]</td>
<td>2013</td>
<td>120</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Stanford Dogs [95]</td>
<td>2011</td>
<td>120</td>
<td>148</td>
<td>159.5</td>
</tr>
<tr>
<td>CUB 2010 [22, 193]</td>
<td>2010</td>
<td>200</td>
<td>20</td>
<td>31</td>
</tr>
<tr>
<td>Oxford Flowers 103 [139]</td>
<td>2008</td>
<td>102</td>
<td>40</td>
<td>66</td>
</tr>
<tr>
<td>Oxford Flowers 17 [137, 140]</td>
<td>2006</td>
<td>17</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Birdsnap [15]</td>
<td>2014</td>
<td>500</td>
<td>69</td>
<td>100</td>
</tr>
<tr>
<td>NABirds [179]</td>
<td>2015</td>
<td>555</td>
<td>13</td>
<td>91</td>
</tr>
<tr>
<td>CompCars [203]</td>
<td>2015</td>
<td>1,631</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>iNaturalist [180]</td>
<td>2017</td>
<td>5,089</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>Columbia Dogs [119]</td>
<td>2015</td>
<td>133</td>
<td>33</td>
<td>62</td>
</tr>
<tr>
<td>Leafsnap [102]</td>
<td>2012</td>
<td>185</td>
<td>56</td>
<td>162</td>
</tr>
<tr>
<td>Leeds Butterflies [56]</td>
<td>2009</td>
<td>10</td>
<td>55</td>
<td>88.5</td>
</tr>
<tr>
<td>Baidu Dogs-in-the-Wild [169]</td>
<td>2018</td>
<td>362</td>
<td>89</td>
<td>212.5</td>
</tr>
</tbody>
</table>

of the 136 papers we collected with publication time between 2008 - 2020, 98 of them report results on the CUB-200-2011 dataset. Joining the CUB-200 dataset, other datasets that are commonly used for comparison include the Stanford Cars [98] and FGVC Aircraft [124, 181] datasets.

While CUB-200-2011 is the clear standard, in an effort to be comprehensive, we have included almost all known datasets that cover fine-grained visual domains. Example domains include: flowers, leaves, butterflies, birds, dogs, cars, airplanes. Datasets for face recognition and human identity maintenance are more appropriately classified within Biometrics and are intentionally excluded from this review (see [20, 88, 125, 170] for detailed coverage of work and datasets related to Biometrics). While some early fine-grained datasets are more than a decade old and have fewer than 10 categories, more recent datasets have hundreds of categories and/or hundreds of images per category. One of the latest fine grained dataset, iNat 2017 [180], features 5,089 different categories with more than 675,000 total training/validation images.
Table 6.2: Results on the CUB-200-2010 Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murray[136]</td>
<td>17.2</td>
<td>Goring[74]</td>
<td>35.9</td>
</tr>
<tr>
<td>Welinder[193]</td>
<td>19.0</td>
<td>Farrell[57]</td>
<td>37.1</td>
</tr>
<tr>
<td>Chen[38]</td>
<td>19.2</td>
<td>Xu[202]</td>
<td>40.2</td>
</tr>
<tr>
<td>Yao[206]</td>
<td>19.2</td>
<td>Kanan[92]</td>
<td>42.8</td>
</tr>
<tr>
<td>Parkhi[144]</td>
<td>22.4</td>
<td>Yao[207]</td>
<td>44.7</td>
</tr>
<tr>
<td>Chai[34]</td>
<td>23.3</td>
<td>Chai[37]</td>
<td>47.3</td>
</tr>
<tr>
<td>Chai[36]</td>
<td>26.7</td>
<td>Zhang[216]</td>
<td>57.4</td>
</tr>
<tr>
<td>Yang[204]</td>
<td>28.2</td>
<td>Parkhi[145]</td>
<td>6.9</td>
</tr>
<tr>
<td>Pu[147]</td>
<td>28.4</td>
<td>Donahue[49]</td>
<td>65.0</td>
</tr>
<tr>
<td>Angelova[8]</td>
<td>30.2</td>
<td>Lin[112]</td>
<td>65.2</td>
</tr>
<tr>
<td>Angelova[7]</td>
<td>30.5</td>
<td>Branson[22]</td>
<td>66.0</td>
</tr>
<tr>
<td>Deng[45]</td>
<td>32.8</td>
<td>Zhang[213]</td>
<td>66.1</td>
</tr>
<tr>
<td>Zhang[217]</td>
<td>34.5</td>
<td>Engin[55]</td>
<td>86.5</td>
</tr>
<tr>
<td>Angelova[6]</td>
<td>34.9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There is some variation in the training regimes specified amongst the different datasets. Most of the datasets include a canonical training/test split, which divides the dataset into two sets of images, one set to train models on and the other to quantify performance with. The most common training approach uses just the images of the training set, together with their respective category labels. Another common approach allows usage of provided object bounding boxes, enabling models that are able to, for example, focus their learning on the object while ignoring the background. While these two training regimes are by far the most common, several recognition approaches also leverage additional annotations such as keypoint part locations or object segmentations. For evaluation of the test set, the images in isolation are most common; however, occasionally, bounding boxes or other annotations are assumed to be known at test time.

The most common performance metric is mean class accuracy, which calculates the average accuracy for each testing class. This simple measure can be good for head-to-head
### Table 6.3: Results on Flowers-102 Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parkhi[145]</td>
<td>53.7</td>
<td>Chai[36]</td>
<td>85.6</td>
</tr>
<tr>
<td>Parkhi[144]</td>
<td>73.3</td>
<td>Sharif[159]</td>
<td>86.8</td>
</tr>
<tr>
<td>Chai[34]</td>
<td>80.0</td>
<td>Zhang[212]</td>
<td>89.3</td>
</tr>
<tr>
<td>Angelova[6]</td>
<td>80.4</td>
<td>Qian[148]</td>
<td>89.5</td>
</tr>
<tr>
<td>Angelova[8]</td>
<td>80.7</td>
<td>Xie[198]</td>
<td>95.3</td>
</tr>
<tr>
<td>Xie[198]</td>
<td>82.1</td>
<td>Zhang[214]</td>
<td>95.9</td>
</tr>
<tr>
<td>Chen[38]</td>
<td>82.6</td>
<td>Huang[84]</td>
<td>96.1</td>
</tr>
<tr>
<td>Murray[136]</td>
<td>84.6</td>
<td>Simon[165]</td>
<td>96.7</td>
</tr>
</tbody>
</table>

comparison, but can be affected by several important factors. Model complexity is often neglected when reporting accuracy, and additional data or annotations can be used, making a horizontal comparison unfair without considering the context. Moreover, it can be beneficial to report category-wise accuracy to better understand challenging categories.

### 6.4.1 Dataset Overview

Table 6.2 provides a comprehensive overview of the important fine-grained datasets sorted by usage popularity. CUB-200-2011 is undoubtedly the most widely-used dataset with 98 papers reporting results on it. Other popular datasets include Stanford Cars [98] (44 reports), FGVC Aircraft [124, 181] (37 reports), Stanford Dogs [95] (37 reports), CUB-200-2010 [22, 193] (27 reports) and Oxford Flowers [139] (18 reports), etc.

For the above 6 popular datasets, we put up a comprehensive performance comparison for papers using the same dataset in Table 6.2 6.4 6.3 6.5 6.6 6.7 sorted by their final accuracy. Figure 6.8 6.9 6.10 6.11 6.12 6.13 shows how mean category accuracy grows with time. Although the performance has been increasing steadily, it seems to have plateaued recently that the performance gain has been within 1% for many state-of-the-art methods.
Figure 6.8: Comparison of Performance over time on the 2010 version of the Caltech-UCSD Birds [193] Dataset (CUB-200-2010). The horizontal and vertical axes indicate time and accuracy, respectively.

Figure 6.9: Comparison of Performance over time on the 2011 version of the Caltech-UCSD Birds [183] Dataset (CUB-200-2011). The horizontal and vertical axes indicate time and accuracy, respectively.
Figure 6.10: Comparison of Performance over time on the Stanford Cars [98] Dataset. The horizontal and vertical axes indicate time and accuracy, respectively.

Figure 6.11: Comparison of Performance over time on the Stanford Dogs [95] Dataset. The horizontal and vertical axes indicate time and accuracy, respectively.
Figure 6.12: Comparison of Performance over time on the Aircrafts [124] Dataset. The horizontal and vertical axes indicate time and accuracy, respectively.

Figure 6.13: Comparison of Performance over time on the Oxford-102 Flowers [139] Dataset. The horizontal and vertical axes indicate time and accuracy, respectively.
6.4.2 Workshop Competitions

Together with ongoing evaluation of new techniques on the published fine-grained datasets, another venue for benchmarking performance is via competitions. Such competitions are usually held at two workshops in conjunction with major computer vision conferences, with the competition running during the months preceding the conference. The FGComp 2013 fine-grained challenge was held in July-November 2013 and the results were presented at the ILSVRC13 workshop at ICCV 2013. The iNaturalist and iMaterialist competitions, which began in April and May 2017 concluded with the results being presented at the FGVC4 workshop [4] at CVPR 2017. In the pursuing years, the FGVC community witnessed a boost in both solution quality and competition quantity: 17 competitions have been held and 214 teams have participated in iNaturalist 2019 competition.

The FGComp Challenge was a composite benchmark combining five different domains: *aircraft, birds, cars, dogs,* and *shoes*. These domains respectively contained 100, 83, 196, 120 and 70 categories, 649 categories in all, with an average of 75 images per category. Challenge participants developed their classification approaches and then evaluated them on each domain independently. Predictions were submitted, and the mean class accuracy was used.
for comparison with the other teams. The overall winning entry was submitted by Gosselin, Murray, Jégou and Perronnin, a joint team from INRIA/XEROX; their fisher-vector-based approach attained an average accuracy of 77.07%.

The iNaturalist Challenge 2017 [1] was a fine-grained challenge sponsored by Google that is an order of magnitude larger than FGComp. With image data provided by iNaturalist.org, this challenge provided 675,170 training/validation images across 5089 categories. These categories span a diverse taxonomic subset of the natural world including plants, fungi, insects, spiders, birds, amphibians, mollusks, mammals, etc. Where FGComp evaluated each domain independently, the iNat competition requires that the prediction for each image be made considering the entire taxonomic spectrum (all domains) at once. The metric used for comparison between participating teams is top-1 error. The winning team GMV attains 0.04809 in public score and 0.04875 in private score.

Alongside the iNaturalist challenge, Google presented the iMaterialist Challenge 2017, which focused on multi-label/attribute prediction in the space of apparel/fashion categories. Specific verticals utilized in the competition include four apparel classes (outerwear, dresses, pants, shoes) and 381 attribute labels. The results are evaluated on top-1 error rate in the predicted labels.
Table 6.6: Results on the CUB-200-2011 Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Method</th>
<th>Accuracy</th>
<th>Method</th>
<th>Accuracy</th>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wah[183]</td>
<td>10.3</td>
<td>Zhang[218]</td>
<td>76.4</td>
<td>Jaderberg[87]</td>
<td>84.1</td>
<td>Sun[169]</td>
<td>86.5</td>
</tr>
<tr>
<td>Murray[136]</td>
<td>33.3</td>
<td>Niu[141]</td>
<td>76.5</td>
<td>Kong[96]</td>
<td>84.2</td>
<td>Simon[165]</td>
<td>86.5</td>
</tr>
<tr>
<td>Lee[108]</td>
<td>41.0</td>
<td>Huang[85]</td>
<td>76.6</td>
<td>Zhang[222]</td>
<td>84.5</td>
<td>Dubey[52]</td>
<td>86.5</td>
</tr>
<tr>
<td>Elhoseiny[54]</td>
<td>43.6</td>
<td>Zhang[223]</td>
<td>77.1</td>
<td>Xu[201]</td>
<td>84.6</td>
<td>Dubey[51]</td>
<td>86.9</td>
</tr>
<tr>
<td>Liu[118]</td>
<td>44.1</td>
<td>Xu[202]</td>
<td>77.4</td>
<td>Zhang[223]</td>
<td>84.7</td>
<td>Sharma[160]</td>
<td>86.9</td>
</tr>
<tr>
<td>Pu[147]</td>
<td>44.2</td>
<td>Ge[67]</td>
<td>77.5</td>
<td>Lin[115]</td>
<td>85.1</td>
<td>Yu[211]</td>
<td>87.1</td>
</tr>
<tr>
<td>Zhang[217]</td>
<td>46.8</td>
<td>Zhao[227]</td>
<td>79.0</td>
<td>Zhang[213]</td>
<td>85.1</td>
<td>Huang[86]</td>
<td>87.3</td>
</tr>
<tr>
<td>Gosselin[75]</td>
<td>49.8</td>
<td>Zhang[225]</td>
<td>79.3</td>
<td>Cai[31]</td>
<td>85.3</td>
<td>Wang[189]</td>
<td>87.4</td>
</tr>
<tr>
<td>Xie[198]</td>
<td>50.8</td>
<td>Yao[209]</td>
<td>79.8</td>
<td>Simon[164]</td>
<td>85.3</td>
<td>Lam[105]</td>
<td>87.5</td>
</tr>
<tr>
<td>Branson[27]</td>
<td>56.5</td>
<td>Zhou[233]</td>
<td>80.4</td>
<td>Liu[122]</td>
<td>85.4</td>
<td>Liang[110]</td>
<td>87.5</td>
</tr>
<tr>
<td>Goring[74]</td>
<td>57.8</td>
<td>Cui[41]</td>
<td>80.7</td>
<td>Branson[26]</td>
<td>85.4</td>
<td>Zhang[224]</td>
<td>87.6</td>
</tr>
<tr>
<td>Freytag[60]</td>
<td>58.5</td>
<td>Xie[198]</td>
<td>81.0</td>
<td>Liu[122]</td>
<td>85.5</td>
<td>Luo[123]</td>
<td>87.7</td>
</tr>
<tr>
<td>Chai[37]</td>
<td>59.4</td>
<td>Ge[68]</td>
<td>81.1</td>
<td>He[77]</td>
<td>85.5</td>
<td>Chen[39]</td>
<td>87.8</td>
</tr>
<tr>
<td>Freytag[59]</td>
<td>60.8</td>
<td>Lin[114]</td>
<td>81.3</td>
<td>He[78]</td>
<td>85.7</td>
<td>Ji[89]</td>
<td>88.1</td>
</tr>
<tr>
<td>Gavves[64]</td>
<td>62.7</td>
<td>Yao[210]</td>
<td>81.6</td>
<td>Lin[114]</td>
<td>85.8</td>
<td>Li[109]</td>
<td>88.7</td>
</tr>
<tr>
<td>Pang[143]</td>
<td>64.6</td>
<td>Tang[175]</td>
<td>82.0</td>
<td>Wei[192]</td>
<td>85.8</td>
<td>Wang[190]</td>
<td>88.8</td>
</tr>
<tr>
<td>Sharif[159]</td>
<td>65.0</td>
<td>Xu[200]</td>
<td>82.8</td>
<td>Wang[186]</td>
<td>86.2</td>
<td>Guo[76]</td>
<td>89.2</td>
</tr>
<tr>
<td>Xie[197]</td>
<td>66.3</td>
<td>Krause[99]</td>
<td>82.8</td>
<td>Moghim[133]</td>
<td>86.2</td>
<td>Cui[43]</td>
<td>89.3</td>
</tr>
<tr>
<td>Xiao[195]</td>
<td>69.7</td>
<td>Yao[208]</td>
<td>82.9</td>
<td>Lin[113]</td>
<td>86.2</td>
<td>Zhang[214]</td>
<td>89.4</td>
</tr>
<tr>
<td>Zhou[232]</td>
<td>70.5</td>
<td>Wang[185]</td>
<td>83.0</td>
<td>Cui[42]</td>
<td>86.2</td>
<td>Cui[43]</td>
<td>89.6</td>
</tr>
<tr>
<td>Gosselin[75]</td>
<td>71.7</td>
<td>Zhang[219]</td>
<td>83.0</td>
<td>Liang[111]</td>
<td>86.3</td>
<td>Du[50]</td>
<td>89.6</td>
</tr>
<tr>
<td>Berg[13]</td>
<td>73.3</td>
<td>Huang[84]</td>
<td>83.3</td>
<td>Zhang[220]</td>
<td>86.3</td>
<td>Ge[66]</td>
<td>90.4</td>
</tr>
<tr>
<td>Ge[68]</td>
<td>73.4</td>
<td>Huang[84]</td>
<td>83.7</td>
<td>Zheng[229]</td>
<td>86.5</td>
<td>Krause[100]</td>
<td>92.3</td>
</tr>
<tr>
<td>Peng[146]</td>
<td>74.8</td>
<td>Gao[63]</td>
<td>84.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ideally, these competitions would have subsequently become benchmarks used to evaluate and compare newly proposed FGVC techniques, much as the annual PASCAL VOC Competitions [2] held in 2005-2012 did for object detection. The recent iNaturalist challenge will likely become a standard benchmark used to evaluate future approaches for fine-grained recognition, alongside datasets such as CUB-200.
Table 6.7: Results on Stanford Dogs Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Method</th>
<th>Accuracy</th>
<th>Method</th>
<th>Accuracy</th>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chai[36]</td>
<td>26.9</td>
<td>Gavves[64]</td>
<td>50.1</td>
<td>Sermanet[157]</td>
<td>76.8</td>
<td>Fu[61]</td>
<td>87.3</td>
</tr>
<tr>
<td>Akata[5]</td>
<td>35.1</td>
<td>Zhang[215]</td>
<td>50.8</td>
<td>Huang[84]</td>
<td>78.3</td>
<td>Cui[43]</td>
<td>88.0</td>
</tr>
<tr>
<td>Yang[204]</td>
<td>38.0</td>
<td>Gosselin[75]</td>
<td>52.9</td>
<td>Zhang[225]</td>
<td>80.4</td>
<td>Liang[111]</td>
<td>88.1</td>
</tr>
<tr>
<td>Pu[147]</td>
<td>39.3</td>
<td>Zhang[221]</td>
<td>53.5</td>
<td>Kranse[100]</td>
<td>80.8</td>
<td>Luo[123]</td>
<td>88.9</td>
</tr>
<tr>
<td>Angelova[7]</td>
<td>39.5</td>
<td>Xie[198]</td>
<td>68.6</td>
<td>Zhao[227]</td>
<td>81.5</td>
<td>Zhang[212]</td>
<td>90.7</td>
</tr>
<tr>
<td>Chai[37]</td>
<td>45.6</td>
<td>Qian[148]</td>
<td>70.3</td>
<td>Dubey[52]</td>
<td>83.6</td>
<td>Rodriguez[151]</td>
<td>92.9</td>
</tr>
<tr>
<td>Gosselin[75]</td>
<td>45.7</td>
<td>Xu[202]</td>
<td>71.4</td>
<td>Niu[141]</td>
<td>85.2</td>
<td>Engin[55]</td>
<td>93.2</td>
</tr>
<tr>
<td>Kanan[92]</td>
<td>47.7</td>
<td>Zhang[222]</td>
<td>72.0</td>
<td>Sun[169]</td>
<td>85.2</td>
<td>Zhang[224]</td>
<td>96.7</td>
</tr>
<tr>
<td>Xie[199]</td>
<td>49.4</td>
<td>Zhang[220]</td>
<td>74.5</td>
<td>Wang[186]</td>
<td>87.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.4.3 Category-wise Accuracy

While these performance comparisons provide a gestalt sense of which algorithms perform most highly, they fail to shed light on what types of input images a given algorithm might do better (or worse) on than its peers. Such insights simply cannot be gleaned from scalar accuracy metrics.

Toward this goal of more detailed insight and understanding, we have begun to collect the actual per-image predictions of the various algorithms from their respective authors.

At present, we have gathered this data as a resource to the community, hopeful to subsequently publish detailed insights about the various algorithms and the datasets themselves.

6.5 FGVC Workshops and FGVC.org Website

FGVC workshops, held in conjunction with major computer vision conferences like CVPR and ICCV, have become a major venue for the exchange of innovative ideas in FGVC. Seven FGVC workshops have been held from 2011 to 2020 with around 10-20 posters presented at each workshop; each workshop has accelerated research and collaboration in this field. Researchers and domain experts from different backgrounds, including computer vision,
psychology, biology, museum curators and industry, are invited to deliver talks at each workshop.

An integral part of the effort undertaken in producing this survey was the creation of the fgvc.org website (click here). This website provides the following information:

- **Workshops** - Links to all past and current FGVC workshops
- **Datasets** - Links to FGVC-related datasets, including those featured in this article (see Section 6.4), are included.
- **Leaderboards** - Along with information about each dataset, a leaderboard is provided. Each leaderboard features (i) a plot of performance over time (similar to *e.g.* Figure 6.9, but *interactive*) and (ii) a table listing all relevant published results, akin to the respective tables in this article (*e.g.* Table 6.6 and the corresponding full table in the supplemental material). These leaderboards will be maintained in perpetuity, continually adding the results of new approaches as they are published.
- **Papers** - A comprehensive list of FGVC-related conference papers, journal articles and dissertations is provided. The entry for each paper includes: bibliographic information (authors, title, venue, *etc.*); a doi link to the publisher’s official webpage for the paper; and, a link to the official PDF version of the paper. The website also allows users to quickly download a bibtex file for any desired subset (or all) of the included papers (this file is generated on demand).
- **Results Files** - We have made a significant effort to contact the authors of individual papers, asking them to provide results files for their published experimental results. These results files allow other researchers in the community to more effectively reproduce published results. Moreover, instead of just seeing the scalar aggregate accuracy across an entire dataset, researchers are able to analyze and understand which categories, and even images, a given approach performs especially well, or conversely, poorly, on. These received result files will be provided timely on the website.
6.6 Conclusion and Future Directions

Much innovative work has been completed in recent years and significant progress has been realized in the FGVC field. Although we divide the proposed algorithms into six common themes, recent papers have explored directions largely orthogonal to the early research efforts. There are now well over a dozen large datasets in fine-grained domains such as butterflies, flowers, birds, dogs, cars, and airplanes. Several of these datasets serve as standards for benchmarking new algorithms and approaches in fine-grained recognition. In the future, we expect to see additional highly-influential works that expand the research frontier and enrich our collective knowledge. We believe that future models will be faster and smaller, more interpretable, and be evaluated in great detail on large-scaled fine-grained datasets.
References


[27] 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-


[53] James Duyck, Chelsea Finn, Andy Hutcheon, Pablo Vera, Joaquin Salas, and Sai Ravela. Sloop: A pattern retrieval engine for individual animal identification. *Pattern*


171


Chapter 7

Conclusion

This dissertation consists of a collection of papers on different but closely-related topics: posed-aligned representation for fine-grained visual categorization (FGVC), network interpretation and understanding, a global pooling study, weakly-supervised keypoint discovery, and an FGVC survey. The fundamental difference between FGVC and general image classification task is that subordinate categories usually have larger intra-class variation and smaller inter-class variation. Part-based representations have a natural advantage compared with other representations as they decompose the object into parts, each containing highly-localized discriminative features. A pose-normalized representation further transforms and aligns object parts so that their features are more robust to object pose variation. Different global pooling layers help the underlying convolutional layers learn different types of features and one key finding is that max pooling encourages part-level features. It is therefore necessary to conduct a comprehensive study of different global pooling layers for FGVC. This study also provides inspiration for a novel pooling layer that helps learn keypoint-level features, leading to a model for weakly-supervised keypoint discovery. In part-based/pose aligned representations, weakly-supervised keypoint discovery can automatically find object keypoints using only image labels, eliminating the costly human annotation process.

FGVC has witnessed a transition from traditional hand-crafted features to features learned by neural networks. The end-to-end learning of millions of parameters makes modern networks hard to interpret and understand. It’s worth noting that we seek not to sacrifice performance for interpretability, but to tackle interpretation in pursuit of better model
performance. A comprehensive FGVC survey helps researchers better understand previous achievements in this field and provides insights about promising future directions.

We briefly summarize the contributions of each individual chapter. Chapter 2 proposes a pose-aligned representation for fine-grained recognition that sets state-of-the-art performances on two widely-used bird datasets. Object keypoints are estimated using a fully-convolutional network. Object parts are then cropped and similarity-transformed relative to a pair of keypoints. The part classifier scores are concatenated for final classification using a three-layer MLP model. Chapter 3 presents a general Bayesian algorithm to associate a network filter with the concepts it represents. The filter-attribute relationship can help generate a textual summarization that explains the reasons behind the network’s decision. It also provides understanding of how concepts are distributed among the network’s filters. Chapter 4 conducts a comprehensive experimental study on the influences of the global pooling layers in FGVC. Several interesting findings are made: (i) k-max pooling outperforms all other pooling schemes; (ii) max pooling performs better than average pooling and (iii) max pooling generalizes better than average pooling. Chapter 5 describes a novel model named LMP-Net for weakly-supervised keypoint discovery. A leaky max pooling (LMP) layer is proposed to help the underlying convolutional layers learn keypoint-level features. A filter selection and attention-area masking procedure is performed to ensure the selectivity and diversity of the network filters. A learnable clustering layer is proposed to greedily output keypoint predictions. Chapter 6 presents a comprehensive survey for FGVC. Models and algorithms are grouped by the general type of approach they follow. A comprehensive comparison across published methods is also included for each popular dataset. FGVC workshops and the fgvc.org website are also introduced in the survey.

7.1 Future Directions

Current popular classification models have an inherent design flaw — they don’t encode geometric constraints between their learned features; this makes them essentially learn bag-
of-visual-words features. This flaw may result in noisy filters that activate on visually-similar but geometrically-implausible features. Several pioneering works [1–3] have sought to encode object-part relationships inside a network. Part-based/pose-normalized representations disentangle object appearance from object pose and show promising results in FGVC. Their application in general image categorization, however, has not been actively explored, mainly due to the fact that objects from different basic categories usually do not share parts. A potential future direction is to device a end-to-end learnable pose-normalized representation for general image categorization. The contributions of this dissertation, like weakly-supervised keypoint discovery and pose-aligned part representation, can serve as starting points.

Network visualization is an intuitive way to interpret and understand neural networks, but such visualizations lack semantic meaning and don’t support quantitative analysis. The proposed filter-attribute probability density function serves as a useful tool to quantitatively understand how concepts are encoded in network filters. This is closely related to the research to understand how knowledge is encoded by neurons in cognitive science. Another potential direction would be the pursuit of a clearer understanding as to how the distribution of concepts across network filters affects network performance followed by a principled way to improve networks given the findings.
References

