Weakly-supervised Discriminative Patch Learning via CNN for Fine-grained Recognition

Yaming Wang, Vlad I. Morariu, Larry S. Davis
University of Maryland, College Park, MD 20742, USA
{wym, morariu, lsd}@umiacs.umd.edu

Abstract

Trending research on fine-grained recognition gradually shifts from traditional multistage frameworks to an end-to-end fashion with convolutional neural network (CNN). Many previous end-to-end deep approaches typically consist of a recognition network and an auxiliary localization network trained with additional part annotations to detect semantic parts shared across classes. In this paper, without the cost of extra semantic part annotations, we advance by learning class-specific discriminative patches within the CNN framework. We achieve this by designing a novel asymmetric two-stream network architecture with supervision on convolutional filters and a non-random way of layer initialization. Experimental results show that our approach is able to find high-quality discriminative patches as expected and gets comparable results to state-of-the-art on two publicly available fine-grained recognition datasets.

1. Introduction

The task of fine-grained object recognition involves distinguishing sub-categories under the same super-category (*e.g.*, birds [40], dogs [21], cars [24] and aircrafts [32]), and solutions find and utilize the information from localized regions to capture the subtle differences. The most effective methods employ convolutional neural networks (CNN) and can be roughly separated into two categories. The first category is a traditional multistage framework built upon CNN features, which finds discriminative regions or semantic parts and constructs an image-level representation out of them [22, 44, 35, 54, 42]; the second one typically consists of a recognition network assisted by a localization network trained with additional part annotations to detect semantic parts [51, 27, 50, 17, 43].

While recent combinations of multistage framework and CNN features achieves good performances - they not only outperform by a large margin their baseline of finetuning the same CNN used for feature extraction [22, 54] but also find high-quality discriminative regions or parts without part annotations [42] - the multistage nature of the framework limits their potential. This is because some stages in their methods depend on a CNN model pretrained with generic data (*e.g.* ImageNet) without further tuning, which might not be optimal. It is also reported that finetuning the network might even hurt performance in these stages [22].

On the other hand, the second category of approaches [50, 17, 43] depends on extra semantic part annotations, making them more expensive. Also, the training process usually involves separately tuning the localization and recognition networks followed by joint training, and such a multistage training strategy might make the integrated network tricky to tune. Furthermore, the motivation for using semantic parts for classification is to *find the corresponding parts* and then *compare their appearance*. The former requires the semantic parts (*e.g.* head and body of birds) to be shared across object classes, encouraging the representations of the parts to be similar, but the latter encourages the part representations to be different across categories in order to be discriminative. This subtle conflict implies a trade-off between the recognition ability and localization ability, which might make it difficult for a single integrated network to achieve optimal classification performance.

We address the issues of both categories. Our main contribution is to explicitly learn discriminative patches within a CNN framework without part annotations. Conceptually, our discriminative patches differ from semantic parts in that the former are not necessarily shared across classes as long as they have discriminative appearance. Therefore, a trade-off between recognition and localization when using semantic parts is avoided so that the network can fully focus on classification. Technically, regarding a convolutional filter as a patch detector, a discriminative patch only gives a high response at a certain region in one class. In the rest of the paper, we demonstrate that such discriminative filters can be learned through an asymmetric two-stream network architecture with filter supervision and proper layer initialization. The resulting framework learns to find high-quality discriminative patches as well as obtaining good classifica-

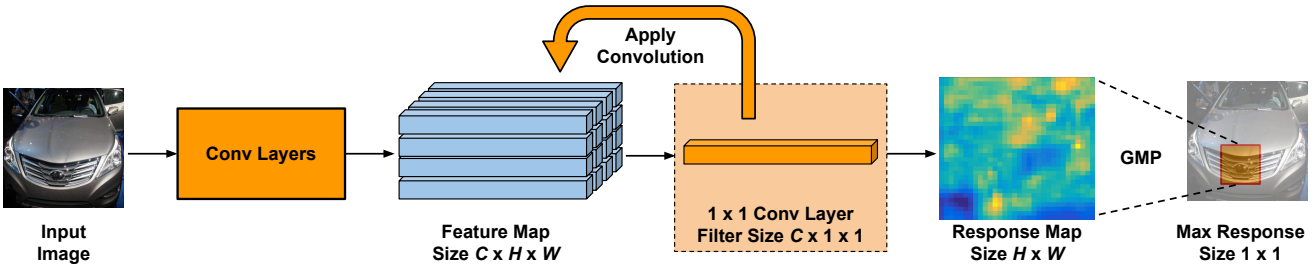

Figure 1. The motivation of our approach is to regard a $C \times 1 \times 1$ vector in a feature map as the representation of a small patch and a 1×1 convolutional filter as a discriminative patch detector. By convolving the feature map with the 1×1 filter, a discriminative patch can be obtained by performing Global Max Pooling (GMP) over the response map.

tion performance; this is achieved in an end-to-end fashion without the cost of semantic part annotations, *i.e.*, it preserves the advantages of both categories.

2. Related Work

Fine-grained recognition Trending research in fine-grained recognition is gradually shifting to deep learning from multistage frameworks [47, 37, 13, 30, 33, 29, 8, 52, 4, 3, 48, 6, 34, 10, 53, 14] based on hand-crafted features such as utilizing DPM [53] or Fisher Vector [14]. As discussed in great details in Section 1, a large portion of current approaches is under the spirit of finding semantic parts or discriminative regions, which can be roughly categorized as "multistage framework with CNN feature" [22, 44, 35, 54, 42] or "localization-recognition integrated network" [51, 27, 50, 17, 43] whose localization network is usually R-CNN and its variants [12, 11] or FCN (Fully Convolutional Network) [31]. Besides these approaches, researchers have explored many other directions such as introducing more effective layers to replace fully-connected layers [28, 9], utilizing label structures [57] or joint embedding spaces of visual (images) and textual(class names) information [55, 2], introducing human in the loop [41, 5, 7], and collecting larger amount of data [45, 46, 23]. Among them, it is worth mentioning that [28] uses a symmetric two-stream network architecture and a bilinear module, *i.e.*, taking the outer product over the outputs of the two streams followed by a series of normalization. [9] further observed that the symmetric two-stream architecture is not necessary and same performance can be achieved by taking the outer product over a single-stream output and itself. Therefore, improvements are actually obtained by replacing the traditional fully-connected layers with this novel bilinear module. In Section 4, we test our network using both the fully-connected layers and the bilinear module for fair comparison.

Intermediate representations in CNN Via effective visualization [49], it is widely-known that the intermediate layers of CNN learn human-interpretable patterns from edges and corners to parts and objects. Regarding the discriminativeness of such patterns, there are two hypothesis. The first is that some neurons in these layers behave as "grandmother cells" which only fire at certain categories, and the second is that the neurons forms a distributed code where the firing pattern of a single neuron is not distinctive and the discriminativeness is distributed among all the neurons. As empirically observed by [1], classical CNN learns a combination of "grandmother cells" and a distributed code. This observation is further supported by [56], which found that by taking proper weighted average over all the feature maps produced by a convolutional layer, one can effectively visualize all the regions in the input image used for classification. Note that both [1] and [56] are based on the original CNN structure and the quality of representation learning remains the same or slightly worse for the sake of better localization. On the other hand, [26, 19, 20] aimed to learn more discriminative representations by putting supervision on intermediate layers, usually by transforming the layer output through a fully-connected layer followed by a loss layer. These works more or less adopt a theoretical perspective which, to some degree, makes their methods difficult to visualize. In contrast, the effectiveness of our approach is very easy to visualize since we regard the convolutional filters as patch detectors from an intuitive perspective. Detailed comparison can be found in Section 3.3

3. Learning Discriminative Patch Detectors within CNN

3.1. Motivation and Overview

Our discriminative patch learning CNN framework regards a 1×1 convolutional filter as a small patch detector. Specifically, referring to Figure 1, if we pass an input image through a series of convolutional and pooling layers to obtain a feature map of size $C \times H \times W$, we can regard each $C \times 1 \times 1$ vector across channels at fixed spatial location as the representation of a small patch at a corresponding location in the original image. Suppose we have learned a

1×1 filter which has high response at a certain discriminative region; by convolving the feature map with this filter we obtain a heatmap. Therefore the discriminative patch can be found simply by picking the location with the maximum value in the entire heatmap. The operation of spatially pooling the entire feature map into a single value is defined as Global Max Pooling (GMP) [56].

Practically, two requirements for the feature map are needed to make this idea suitable for fine-grained recognition. Firstly, since the discriminative regions in fine-grained categories are usually highly localized, we need a relatively small receptive field, *i.e.*, each $C \times 1 \times 1$ vector represents a relatively small patch in the original image. Secondly, since fine-grained recognition involves accurate localization of these regions, the corresponding stride in the original image between adjacent patches should also be small. In earlier network architectures, the size and stride of the convolutional filters and pooling kernels are large. As a result, the receptive field of a single neuron in later convolutional layers is large, so is the stride for adjacent fields. For example, in AlexNet [25], the minimum receptive field of the 5^{th} convolutional layer `conv5` is as large as 197×197 with stride 32, which is not fine enough for the task. Fortunately, the evolution of network architectures [36] [38] [16] is making the filter size and pooling kernel smaller as the networks go deeper. For example, in a 16-layer VGG network (VGG-16), the output of the 10^{th} convolutional layer can represent patch as small as 92×92 with stride 8, which is small and dense enough for our task given a standard original image size of 256×256.

In the rest of Section 3, we will demonstrate how a set of discriminative patch detectors can be effectively learned as a 1×1 convolutional layer in a network specifically designed for this task. An overview of our framework is displayed in Figure 2, which is based on VGG-16. There are three key components in our design: the asymmetric two-stream structure to learn discriminative patches as well as global features (Section 3.2), the convolutional filter supervision to ensure the discriminativeness of the patch detectors (Section 3.3) and the non-random layer initialization to accelerate the network convergence (Section 3.4). Note that, though we use VGG-16 to illustrate our approach, the ideas are not limited by the network architecture.

3.2. Asymmetric Two-stream Architecture

The core component of the network responsible for discriminative patch learning is a 1×1 convolutional layer followed by a GMP layer, as displayed in Figure 1 and discussed in detail in Section 3.1. The component followed by a classifier (*e.g.*, several fully-connected layers and a softmax layer) forms the discriminative patch stream (P-Stream) of our network. The P-Stream uses the output of `conv4-3` and the minimum receptive field in this feature map corresponds to a patch of size 92×92 with stride 8.

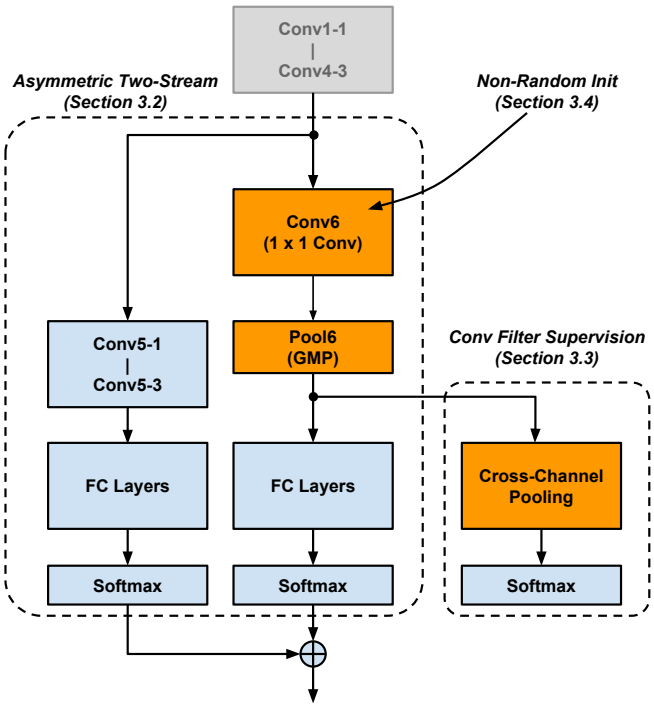

Figure 2. Overview of our framework, which consists of a) an asymmetric two-stream architecture to learn both the discriminative patches and global features, b) supervision imposed to learn discriminative patch detectors and c) non-random layer initialization. For simplicity, except GMP, all pooling and ReLU layers between convolutional layers are not displayed.

In practice, however, the recognition of some fine-grained categories might depend more on global shape and appearance instead of a few discriminative patches. To give the network flexibility to learn global shape and appearance, in another stream we preserve the further convolutional and pooling layers followed by a classifier. The last pooling layer, then, has a minimum receptive field of 212×212, which is almost as large as 224×224 network intput cropped out of a 256×256 image and represent a set of global features. Therefore, this stream focuses more on global features; and we refer to it as the G-Stream. We merge the two streams in the end.

3.3. Convolutional Filter Supervision

Using the network architecture described above, the 1×1 convolutional layer in P-Stream is not guaranteed to fire at discriminative patches as desired. For the framework to learn class-specific discriminative patch detectors, we impose supervision directly at the 1×1 filters by introducing a Cross-Channel Pooling layer followed by a softmax loss layer, the details of which are displayed in Figure 3 and the

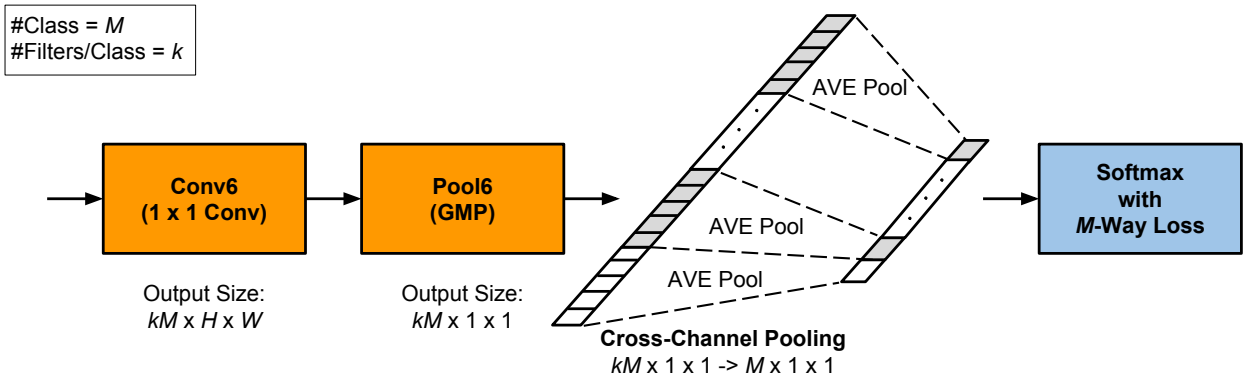

Figure 3. The illustration of our convolutional filter supervision. The filters in `conv6` are grouped into M groups, where M is the number of classes. The maximum responses in group i are averaged into a single score indicating the effect of the discriminative patches in Class i. The pooled vector is fed into a softmax loss layer to encourage discriminative patch learning.

integration of which into the whole framework is shown in Figure 1.

The filter supervision works as follows. Suppose we have M classes and each class has k discriminative patch detectors; then the number of 1×1 filters required is kM. After obtaining the max response of each filter through GMP, we get a kM-dimensional feature vector. By Cross-Channel Pooling, we average the values of this vector from dimension $(ki + 1)$ to dimension $(k + 1)i$ as the averaged response of discriminative patch detectors from Class $(i + 1)$, resulting in an M-dimensional vector. By feeding the pooled vector into an M-class softmax loss, we encourage the filters from any class to find discriminative patches from training samples of that class, such that their averaged filter response is large. The reason to use average pooling instead of max pooling is that we want all the filters from a given class to have balanced responses. Average pooling tends to affect all pooled filters during back propogation, while max pooling only affects the filter with the maximum response. Similar considerations are discussed in [56].

Using this form of supervision, since there is no learnable parameter between the softmax loss and the 1×1 convolutional layer, by taking the partial derivatives of the loss w.r.t. the filter weights, we can directly adjust the filter weights via the loss function. We believe this is a key difference from previous approaches which introduce intermediate supervision [26, 19, 20]. Unlike us, [26, 19, 20] have learnable weights (usually a fully-connected layer) between the side loss and the main network, which essentially learn the weights of a classifier unused at test time. The main network is only effected by back-propogating the gradients of these weights. In our approach, the loss directly effects the main network and its effect is much easier to visualize by checking the top patches found by these filters.

3.4. Layer Initialization

In Section 3.3, the side loss can directly effect the 1×1 convolutions. In practice, we found that if the 1×1 convolutional layer is initialized randomly, it converges to bad local minima. For example, the output vector of the Cross-Channel Pooling might approach all-zero to reduce the side loss during training, which is useless. To overcome this problem, we introduce a method for non-random initialization.

The non-random initialization is motivated by our interpretation of a 1×1 filter as a patch detector. The patch detector of Class i is initialized by patch representations from the samples in that class. This is done in a weakly-supervised way without part annotations. Concretely, we extract the `conv4-3` features from the ImageNet pretrained model and compute the energy at each spatial location (*i.e.*, l_2 norm of each $C \times 1 \times 1$ vector in a feature map). As can be seen from the first row of Figure 6, though not perfect, the heatmap of energy distribution acts as a reasonable indicator of useful patches. After choosing non-overlapping high-energy regions from training samples from Class i, we perform k-means clustering over the representations of the chosen patches and pick the cluster centers as the initialization for filters from Class i. To increase their discriminativeness, we further whiten the initializations using the technique from [15] and do l_2 normalization. In practice this simple method provides reasonable initializations which are further refined during end-to-end training. Also, in Section 4 we will see that the energy distribution used for initialization becomes much more discriminative after training.

As long as the layer is properly initialized, the whole network can be trained in an end-to-end fashion just once, which is more efficient compared with the multistage training strategy of previous works [27, 50, 17].

4. Experiments

4.1. Datasets

Stanford Cars dataset [24] has 16,185 images from 196 classes. We follow the standard data split of 8,144 training images and 8,041 test images provided by [24], where each class has approximately 40 training images and 40 test images. No part annotations are provided in this dataset.

CUB-200-2011 dataset [40] has 11,788 images of 200-class fine-grained bird species. We follow the standard data split of 5,994 training images and 5,794 test images provided by [40], where each class has roughly 30 training images and 30 test images. Part annotations are provided but unused in our experiments. Note that recent work [23] obtains the best result of 92.8% on test set with off-the-shelf 42-layer Inception V3 [39], using (Class Name, Image) pairs filtered from 5 million Google Image Search results such that each class has 800 more training samples on average. Since our goal is to demonstrate that our method works for both rigid and non-rigid fine-grained domains, we follow the original training set in our experiments.

4.2. Implementation Details

Network Structure As displayed in Figure 2, our network is based on VGG-16 [36], which has 16 layers with learnable parameters. Regarding the input, we crop an image to its bounding box, resize it to 256×256 and feed a random crop of 224×224 into the network. Notice that this is a relatively economic setting, as recent end-to-end approaches have input size of 448×448 [28, 9] or 800×600 [50] or multi-scales [17]. Discriminative patches are learned via the 1×1 convolution (denoted as `conv6`) layer following `conv4-3`. The size ($C \times H \times W$) of `conv4-3` output is $512 \times 28 \times 28$, therefore the size of each convolutional filter is $512 \times 1 \times 1$. We set the number of filters per class to be 10, resulting in a total of 1960 filters for cars and 2000 for birds. During Cross-Channel average pooling, the maximum responses of each group of 10 filters are pooled into one dimension. The network is built and trained with Caffe [18].

Layer Initialization As discussed in Section 3.4, to initialize `conv6`, we extract `conv4-3` features using ImageNet pretrained model; each image provides 7 patch representations at locations with highest energy (non-maximum suppression is used). For each class, we perform k-means clustering over the features of all the training samples in that class and their horizontal mirrors with the number of clustering centers set to 15. Since each class only requires 10 filter initializations, we select the top 10 centers with the largest number of positive samples in their top activations. To fit the scale of the network, we rescale the selected centers such that the norm of each initialized filter is 0.045. Other convolutional layers are initialized from an ImageNet pretrained model directly (compared with "indirect" initialization of `conv6`) and the fully-connected layers are randomly initialized.

Training/Testing Configurations After the layers are properly initialized, a single-stage end-to-end training is conducted. The learning rate is initialized at 10^{-3} and drops by a factor of 10 every 14,000 iterations. The total number of iterations is 30,000 with a batch size of 16. At test time, for each image we average the 10 classification results from the center and 4 corners with their mirrors.

4.3. Results

For convenience, we denote our approach as DPL-CNN, which is an abbreviation for *Discriminative Patch Learning within a CNN*.

We compare our approach with the corresponding baseline, which involves finetuning a VGG-16 network [36] on the datasets. For fair comparison, we cite previously published baseline results [57]. Specifically, the settings in [57] named "VGG-SM/with BBox/MV" is exactly the same as ours; this stands for "VGG-16 with SoftMax, Bounding Boxes and Multi-View testing (*i.e.*, 10 crops testing as discussed in Section 4.2)". Our results using the baseline are similar to theirs but slightly inferior.

Besides the baseline, we compare our approach to the methods with the best results on either dataset using the given amount of data. To the best of our knowledge, the best result on Stanford Cars dataset is reported in [22] (denoted as CoSeg), which is a multistage framework built upon 19-layer VGG-19 features, involving segmentation, part selection, part representation generation and SVM. Interestingly, the current best result on CUB-200-2011 using the original training set is obtained using an end-to-end method [28], which is the VGG-16 based bilinear network discussed in Section 2 (denoted as B-CNN). In addition to having the best results, these two methods serve as ideal references since (*i*) both have been evaluated on both datasets (which in fact is not quite common in current literature); (*ii*) neither of them uses part annotations, but still outperform those using them; (*iii*) the best results of both are obtained when bounding boxes are presented; (*iv*) both methods are VGG-Net based and there is no significant difference in performance between VGG-16 and VGG-19.

The results are displayed in Table 1.

Though the results using our reproduction of the VGG-16 baseline is slightly lower than the one reported in [57], we believe this is within reasonable range since it is already slightly better than the VGG-19 baseline reported in [22]. As can be seen, our method consistently outperforms the baseline by a significant margin even compared with the version in [57]. This is strong evidence that the discriminative patch learning occurred within the network helps

Method	Part	BBox	End-to-End	Stanford Cars (%)	CUB-200-2011 (%)
FT VGG-19 [22]	-	✓	✓	89.0	75.0
CoSeg [22] (VGG-19 Based)	-	✓	-	92.8	82.8
B-CNN [28] (VGG-16 Based)	-	✓	✓	91.3	85.1
FT VGG-16 [57] (Baseline)	-	✓	✓	89.8	79.8
FT VGG-16 (Our Baseline)	-	✓	✓	89.1	79.0
DPL-CNN (Ours)	-	✓	✓	92.3	83.1

Table 1. The experimental results on both Stanford Cars and CUB-200-2011 datasets. Performance is measured by the accuracy over test set.

the final classification. Another observation is that it is almost equally effective for rigid (cars) and non-rigid (birds) fine-grained objects. One explanation is that our (relatively small) discriminative patches are only determined by local appearance, which is robust to deformation. The appearance of semantic parts, in constrast, can change severely due to deformation or pose variation.

The CoSeg [22] method automatically finds a set of parts and the classification is based on an ensemble of classifiers trained on the representations of each part. Therefore, without part annotations, it still follows the "finding parts and comparing appearance" idea discussed in Section 1. For rigid objects like cars, alignment is relatively easy; but for non-rigid object like birds, there is too much appearance variation due to deformation and pose variation, which increases the difficulty of classification based on these deformed parts. This, to some degree, explains why CoSeg outperform all end-to-end network on cars so far, but we outperform it on birds.

The B-CNN [28] method is highly effective on birds, but its advantages is diminished when dealing with rigid objects, so we outperform it on cars. The motivation of [28] is to expect one stream of the network to focus on "where" while the other focuses on "what". During the experiments, the author observed that the role of the two streams are not well separated and their neurons have similar firing patterns.

To summarize, our approach obtains balanced performance on both rigid and non-rigid fine-grained objects that are comparable to state-of-the-art.

4.4. Visualization and Analysis

4.4.1 Visualization Overview

Insights into the behavior of our approach can be obtained by visualizing the effects of `conv6`, the 1×1 convolutional layer. To thoroughly understand its behavior, the visualizations are constructed from three perspectives.

- Visualize patch activations. Since we regard each filter as a discriminative patch detector, we can identify the learned patches by applying `conv6` filters to images and looking at top activations. We will see we do find high-quality discriminative regions.
- Visualize a forward pass. Since the max responses of these filters are directly used for classification, by visualizing the output of `conv6`'s next layer, `pool6`, we will see that it produces discriminative representations which have high responses for certain classes.
- Visualize back propagation. During training, `conv6` can affect its previous layer, `conv4-3`, through back propagation. By comparing the `conv4-3` features before and after training, we will see that the spatial energy distributions of previous feature maps are changed in a discriminative fashion.

4.4.2 Stanford Cars

The visualization of top activations for some of the 1×1 filters is displayed in Figure 4. Unlike previous filter visualizations which pick human interpretable results randomly among the filter activations, we have imposed supervision on `conv6` filters and can identify their top activations with a certain class. From Figure 4, we see that the top activations are very consistent with human perception and cover a diverse of regions. For instance, the 1^{st} filter belonging to Class 1 (AM General Hummer SUV) activates on the squared side windows of an SUV, the 271^{st} filter captures the discriminative head light of Class 27 (BMW 1 Series Coupe), the 1610^{th} filter focuses on the frontal face of Class 160 (Mercedes-Benz 300-Class Convertible) and the 1843^{rd} belonging to Class 184 (Tesla Model S) captures the distinctive tail of this type. The reasons for the network's ability to localize these subtle discriminative regions are that a) 1×1 filters correspond to a small patch detector in original image b) the filter supervision and c) the inclusion of more than $2/3$ of the cluster centers as initialization which promotes diversity.

The visualization of `pool6` features are shown in Figure 5. We plot the averaged representations over all test samples from a certain class. Since we have learned a set of discriminative filters, the representations should have high responses at one class or only a few classes. Figure 5 indicates that our approach works as expected, resulting in peaky layer output which is fed into the classifier consisting of fully-connected layers and a softmax in a forward pass.

Figure 4. The visualization of patch activations in Stanford Cars. For each filter, we visualize its top-2 activations across the dataset. The results are highly consistent with human perception. For example, in the **third column**, the patch focuses on the distinctive exhaust pipe of the race car; in the **last column**, the patch focuses on the black side stripe which are unique to this type. Other examples are interpretated in Section 4.4.2. The size of the original image is 256×256 and the actual patch size is 92×92.

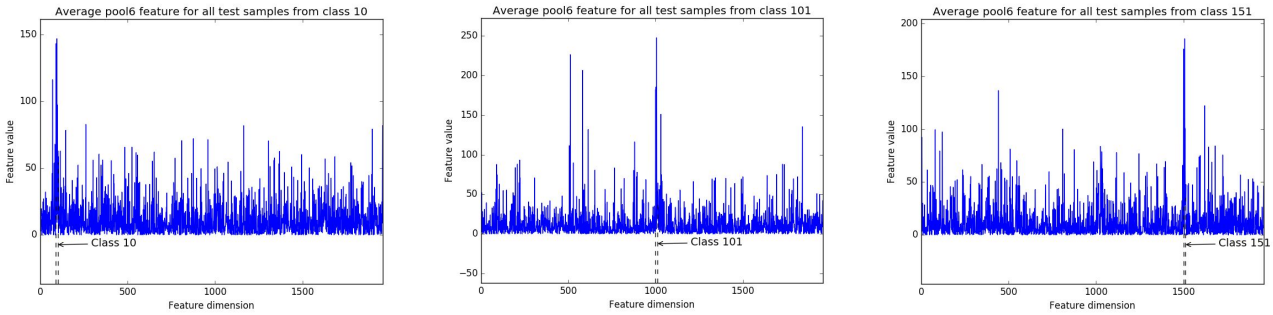

Figure 5. The visualization of the averaged `pool6` features over all test samples from Class 10, 101 and 151 in Stanford Cars. The dash lines indicate the range of values given by the discriminative patch detectors belonging to the class. As can be seen, the representations are peaky at the corresponding class.

Most interesting is the effect of `conv6` on the previous convolutional layer `conv4-3` through back propagation. As discussed in Section 3.4, we use the energy distribution of `conv4-3` as a hint to provide layer initialization. After training, we observed that the energy distribution is refined by `conv6` and becomes more discriminative. Figure 6 provides visualizations of this observation. We map every spatial location in the feature map back to the corresponding patch in the original image, and the value of each pixel is determined by the max energy patch covering that pixel. From the first line of Figure 6, the features extracted from an ImageNet pretrained model tend to have high energy at round patterns such as wheels, some unrelated background shape, a person in the image and some texture patterns, which are common patterns in generic models found in [49]. After training, the energy shifts from these patterns to discriminative regions of cars. For example, in the 2^{nd} column, after training the energy over the brick patterns is reduced. In the 6^{th} column, the feature map has high energy initially at both the wheel and the frontal light; after training, the network has determined that a discriminative patch for that class (Volkswagen Beetle) is the light rather than the wheels. In the 7^{th} column, before training the energy is focused mostly at the air grill, and training adds the discriminative fog light into the high energy region. Therefore, the discriminative patch detectors have beneficial effects on their previous layer during training.

4.4.3 CUB-200-2011

The examples of the discriminative patches found by our approach is displayed in Figure 7, which include the texture and spots with bright color as well as specific shape of beak or webbed. Compared with the visualizations of previous works not using part annotations (*e.g.* [22, 28]), our approach is able to localize such patches more accurately due to the facts that our patch detectors operate over denser and

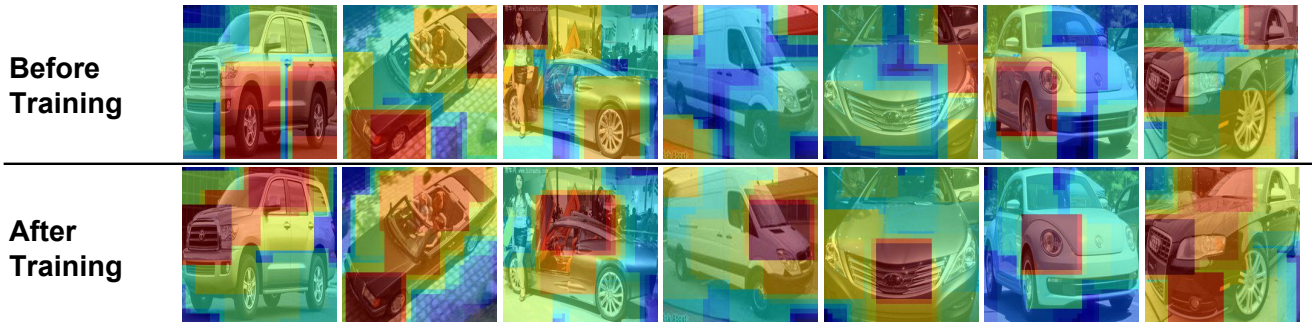

Figure 6. The visualization of the energy distribution of `conv4-3` feature map before and after training for Stanford Cars. We remap each spatial location in the feature map back to the patch in the original image. After training in our approach, the energy distribution is changed in a discriminative fashion. For example, in the **first column**, the high energy region shifts from the wheels to discriminative regions like the frontal face and the top of the vehicle; in the **third column**, the person no longer lies in high energy region after training. More examples are interpretated in Section 4.4.2

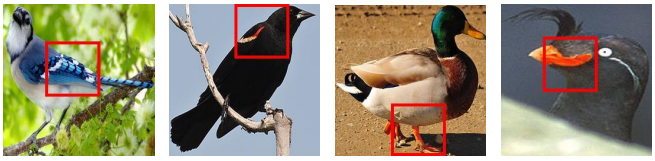

Figure 7. The visualization of patch activations in CUB-200-2011. Our approach is able to accurately localize discriminative patches without part annotations, such as the bright texture in the **first image**, the color spot in the **second image**, the webbed and beak in the **third** and **forth** image, respectively.

smaller patches and are not necessary to be shared across categories.

Similar to cars, the features from the next GMP layers are peaky at certain categories (Figure 8). And the energy distributions of previous convolutional features are effected such that the high energy at background regions like branches is reduced and the discriminative regions become more focused or diversed according to different categories (Figure 9).

5. Conclusion

In this paper, we aim to combine the advantages of previous multistage and end-to-end frameworks for fine-grained recognition. Specifically we learn a set of discriminative patch detectors within a CNN framework in an end-to-end fashion without part annotation, which is done via an asymmetric two-stream network structure with convolutional layer supervision and non-random layer initialization. Experimental results suggest that our approach can learn high-quality discriminative patches as well as obtaining comparable results to state-of-the-art on both rigid and non-rigid fine-grained datasets. Possible future directions include utilizing multi-scale patches represented in different convolutional layers.

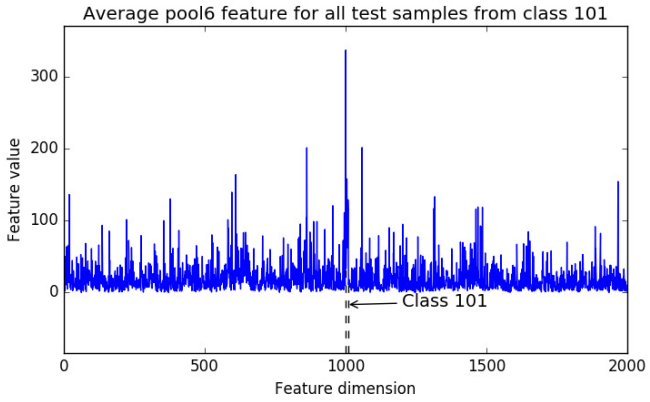

Figure 8. The averaged `pool6` features over all the test samples from Class 101 in CUB-200-2011, which is peaky at corresponding dimensions.

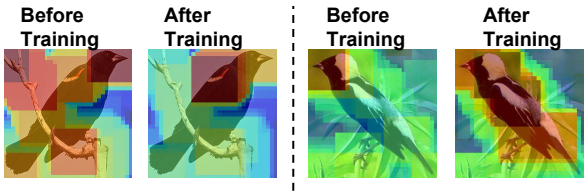

Figure 9. The energy distributions of `conv4-3` feature maps before and after training in CUB-200-2011. After training, in the left example, the high energy region at the background branches is greatly shrinked and the energy is concentrated at the discriminative color spot; in the right example, more energy is distributed to the distinctive black-and-white wing and tail of the species.

Acknowledgements This research was supported in part by funds provided from the Office of Naval Research under grant N000141612713 entitled "Visual Common Sense Reasoning for Multi-agent Activity Prediction and Recognition."

References

[1] P. Agrawal, R. B. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014. 2

[2] Z. Akata, S. E. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, 2015. 2

[3] T. Berg and P. N. Belhumeur. How do you tell a blackbird from a crow? In *ICCV*, 2013. 2

[4] T. Berg and P. N. Belhumeur. POOF: part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In *CVPR*, 2013. 2

[5] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *ECCV*, 2010. 2

[6] Y. Chai, V. S. Lempitsky, and A. Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *ICCV*, 2013. 2

[7] Y. Cui, F. Zhou, Y. Lin, and S. Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *CVPR*, 2016. 2

[8] R. Farrell, O. Oza, N. Zhang, V. I. Morariu, T. Darrell, and L. S. Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *ICCV*, 2011. 2

[9] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *CVPR*, 2016. 2, 5

[10] E. Gavves, B. Fernando, C. G. M. Snoek, A. W. M. Smeulders, and T. Tuytelaars. Fine-grained categorization by alignments. In *ICCV*, 2013. 2

[11] R. B. Girshick. Fast R-CNN. In *ICCV*, 2015. 2

[12] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[13] C. Göring, E. Rodner, A. Freytag, and J. Denzler. Nonparametric part transfer for fine-grained recognition. In *CVPR*, 2014. 2

[14] P. H. Gosselin, N. Murray, H. Jégou, and F. Perronnin. Revisiting the fisher vector for fine-grained classification. *Pattern Recognition Letters*, 49:92–98, 2014. 2

[15] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012. 4

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[17] S. Huang, Z. Xu, D. Tao, and Y. Zhang. Part-stacked cnn for fine-grained visual categorization. In *CVPR*, 2016. 1, 2, 4, 5

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*, pages 675–678, 2014. 5

[19] Z. Jiang, Y. Wang, L. S. Davis, W. Andrews, and V. Rozgic. Learning discriminative features via label consistent neural network. *CoRR*, abs/1602.01168, 2016. 2, 4

[20] X. Jin, Y. Chen, J. Dong, J. Feng, and S. Yan. Collaborative layer-wise discriminative learning in deep neural networks. In *ECCV*, 2016. 2, 4

[21] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. 1

[22] J. Krause, H. Jin, J. Yang, and F. Li. Fine-grained recognition without part annotations. In *CVPR*, 2015. 1, 2, 5, 6, 7

[23] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. The unreasonable effectiveness of noisy data for fine-grained recognition. In *ECCV*, 2016. 2, 5

[24] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representation for fine-grained categorization. In *International IEEE Workshop on 3D Representation and Recognition*, 2013. 1, 5

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3

[26] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015. 2, 4

[27] D. Lin, X. Shen, C. Lu, and J. Jia. Deep LAC: deep localization, alignment and classification for fine-grained recognition. In *CVPR*, 2015. 1, 2, 4

[28] T. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN models for fine-grained visual recognition. In *ICCV*, 2015. 2, 5, 6, 7

[29] Y. Lin, V. I. Morariu, W. H. Hsu, and L. S. Davis. Jointly optimizing 3d model fitting and fine-grained classification. In *ECCV*, 2014. 2

[30] J. Liu, A. Kanazawa, D. W. Jacobs, and P. N. Belhumeur. Dog breed classification using part localization. In *ECCV*, 2012. 2

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 2

[32] S. Maji, E. Rahtu, J. Kannala, M. B. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013. 1

[33] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012. 2

[34] J. Pu, Y. Jiang, J. Wang, and X. Xue. Which looks like which: Exploring inter-class relationships in fine-grained visual categorization. In *ECCV*, 2014. 2

[35] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *ICCV*, 2015. 1, 2

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 3, 5

[37] M. Stark, J. Krause, B. Pepik, D. Meger, J. J. Little, B. Schiele, and D. Koller. Fine-grained categorization for 3d scene understanding. In *BMVC*, 2012. 2

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 3

[39] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, June 2016. 5

[40] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds 200-2011 dataset. In *Technical Report CNS-TR-2011-001, Caltech,*, 2011. 1, 5

[41] C. Wah, G. V. Horn, S. Branson, S. Maji, P. Perona, and S. Belongie. Similarity comparisons for interactive fine-grained categorization. In *CVPR*, 2014. 2

[42] Y. Wang, J. Choi, V. Morariu, and L. S. Davis. Mining discriminative triplets of patches for fine-grained classification. In *CVPR*, 2016. 1, 2

[43] X. Wei, C. Xie, and J. Wu. Mask-cnn: Localizing parts and selecting descriptors for fine-grained image recognition. *CoRR*, abs/1605.06878, 2016. 1, 2

[44] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*, 2015. 1, 2

[45] S. Xie, T. Yang, X. Wang, and Y. Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *CVPR*, 2015. 2

[46] L. Yang, P. Luo, C. C. Loy, and X. Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, 2015. 2

[47] S. Yang, L. Bo, J. Wang, and L. G. Shapiro. Unsupervised template learning for fine-grained object recognition. In *NIPS*, 2012. 2

[48] B. Yao, G. R. Bradski, and L. Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *CVPR*, 2012. 2

[49] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 2, 7

[50] H. Zhang, T. Xu, M. Elhoseiny, X. Huang, S. Zhang, A. Elgammal, and D. Metaxas. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *CVPR*, 2016. 1, 2, 4, 5

[51] N. Zhang, J. Donahue, R. B. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, 2014. 1, 2

[52] N. Zhang, R. Farrell, and T. Darrell. Pose pooling kernels for sub-category recognition. In *CVPR*, 2012. 2

[53] N. Zhang, R. Farrell, F. N. Iandola, and T. Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *ICCV*, 2013. 2

[54] X. Zhang, H. Xiong, W. Zhou, W. Lin, and Q. Tian. Picking deep filter responses for fine-grained image recognition. In *CVPR*, 2016. 1, 2

[55] X. Zhang, F. Zhou, Y. Lin, and S. Zhang. Embedding label structures for fine-grained feature representation. In *CVPR*, 2016. 2

[56] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016. 2, 3, 4

[57] F. Zhou and Y. Lin. Fine-grained image classification by exploring bipartite-graph labels. In *CVPR*, 2016. 2, 5, 6