Learning a Discriminative Filter Bank within a CNN for Fine-grained Recognition

Yaming Wang[1], Vlad I. Morariu[2*], Larry S. Davis[1]
[1]University of Maryland, College Park [2]Adobe Research
{wym, lsd}@umiacs.umd.edu morariu@adobe.com

Abstract

Compared to earlier multistage frameworks using CNN features, recent end-to-end deep approaches for fine-grained recognition essentially enhance the mid-level learning capability of CNNs. Previous approaches achieve this by introducing an auxiliary network to infuse localization information into the main classification network, or a sophisticated feature encoding method to capture higher order feature statistics. We show that mid-level representation learning can be enhanced within the CNN framework, by learning a bank of convolutional filters that capture class-specific discriminative patches without extra part or bounding box annotations. Such a filter bank is well structured, properly initialized and discriminatively learned through a novel asymmetric multi-stream architecture with convolutional filter supervision and a non-random layer initialization. Experimental results show that our approach achieves state-of-the-art on three publicly available fine-grained recognition datasets (CUB-200-2011, Stanford Cars and FGVC-Aircraft). Ablation studies and visualizations are provided to understand our approach.

1. Introduction

Fine-grained object recognition involves distinguishing sub-categories of the same super-category (*e.g.*, birds [39], cars [26] and aircrafts [34]), and solutions often utilize information from localized regions to capture subtle differences. Early applications of deep learning to this task built traditional multistage frameworks upon convolutional neural network (CNN) features; more recent CNN-based approaches are usually trained end-to-end and can be roughly divided into two categories: *localization-classification sub-networks* and *end-to-end feature encoding*.

Previous multistage frameworks utilize low-level CNN features to find discriminative regions or semantic parts, and construct a mid-level representation out of them for classification [24, 44, 35, 41, 53, 42]. These methods achieve better performance compared to two types of baselines: *(i)* they outperform their counterparts with hand-crafted features (*e.g.*, SIFT) by a huge margin, which means that low-level CNN features are far more effective than previous hand-crafted ones; *(ii)* they significantly outperform their baselines which finetune the same CNN used for feature extraction. This further suggests that CNN's ability to learn mid-level representations is limited and still has sufficient room to improve. Based on these observations, end-to-end frameworks aim to *enhance the mid-level representation learning capability of CNN*.

The first category, *localization-classification sub-networks*, consists of a classification network assisted by a localization network. The mid-level learning capability of the classification network is enhanced by the localization information (*e.g.* part locations or segmentation masks) provided by the localization network. Earlier works from this category [51, 29, 50, 18, 43] depend on additional semantic part annotations, while more recent ones [20, 9, 55] only require category labels. Regardless of annotations, the common motivation behind these approaches is to first *find the corresponding parts* and then *compare their appearance*. The first step 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, in order to be discriminative, the latter encourages the part representations to be different across classes. This subtle conflict implies a trade-off between recognition and localization ability, which might reduce a single integrated network's classification performance. Such a trade-off is also reflected in practice, in that training usually involves alternating optimization of the two networks or separately training the two followed by joint tuning. Alternating or multistage strategies complicate the tuning of the integrated network.

The second category, *end-to-end feature encoding* [30, 10, 23, 8, 5], enhances CNN mid-level learning by encoding higher order statistics of convolutional feature maps. The need for end-to-end modeling of higher order statistics became evident when the Fisher Vector encodings of

*The work was done while the author was at University of Maryland.

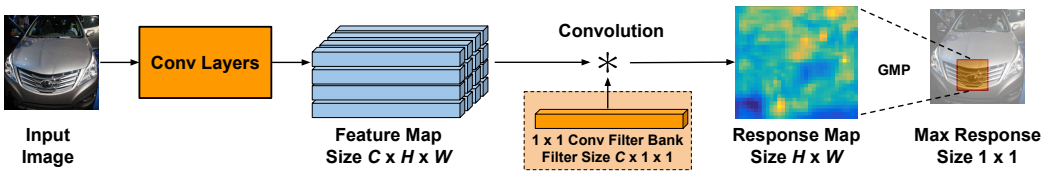

Figure 1. The motivation of our approach is to regard a $C \times 1 \times 1$ vector in a feature map as the representation of a small patch and a 1×1 convolutional filter as a discriminative patch detector. A discriminative patch can be discovered by convolving the feature map with the 1×1 filter and performing Global Max Pooling (GMP) over the response map. The full architecture is illustrated in Figure 2.

SIFT features outperformed a finetuned AlexNet by a large margin on fine-grained recognition [13]. The resulting architectures have become standard benchmarks in the literature. While effective, end-to-end encoding networks are less human-interpretable and less consistent in their performance across non-rigid and rigid visual domains, compared to localization-classification sub-networks.

This paper addresses the issues facing both categories of end-to-end networks. Our main contribution is to explicitly learn discriminative mid-level patches *within* a CNN framework in an end-to-end fashion without extra part or bounding box annotations. This is achieved by regarding 1×1 filters as small "patch detectors", designing an asymmetric multi-stream structure to utilize both patch-level information and global appearance, and introducing filter supervision with non-random layer initialization to activate the filters on discriminative patches. Conceptually, our discriminative patches differ from the parts in localization-recognition sub-networks, such that they are not necessarily shared across classes as long as they have discriminative appearance. Therefore, our network fully focuses on classification and avoids the trade-off between recognition and localization. Technically, a convolutional filter trained as a discriminative patch detector will only yield a high response at a certain region for one class.

The resulting framework enhances the mid-level learning capability of the classical CNN by introducing a bank of discriminative filters. In practice, our framework preserves the advantages of both categories of previous approaches:

- Simple and effective. The network is easy to build and once initialized only involves single-stage training. It outperforms state-of-the-art.
- High human interpretability. This is shown through various ablation studies and visualizations of learned discriminative patches.
- Consistent performance across different fine-grained visual domains and various network architectures.

2. Related Work

Fine-grained recognition Research in fine-grained recognition has shifted from multistage frameworks based on hand-crafted features [52, 3, 48, 6, 13] to multistage framework with CNN features [24, 44, 35, 53, 42], and then to end-to-end approaches. Localization-classification sub-networks [51, 29, 50, 18, 20, 27, 9] have a localization network which is usually a variant of R-CNN [12, 11], FCN (Fully Convolutional Network) [33] or STN (Spatial Transformer Network) [20] and a recognition network that performs recognition based on localization. More recent advances explicitly regress the location/scale of the parts using a recurrent localization network such as LSTM [27] or a specifically designed recurrent architecture [9]. End-to-end encoding approaches [30, 10, 23, 8, 5] encode higher order information. The classical benchmark, Bilinear-CNN [30] uses a symmetric two-stream network architecture and a bilinear module that computes the outer product over the outputs of the two streams to capture the second-order information. [10] further observed that similar performance can be achieved by taking the outer product over a single-stream output and itself. More recent advances reduce high feature dimensionality [10, 23] or extract higher order information with kernelized modules [8, 5]. Others have explored directions such as utilizing hierarchical label structures [57], combining visual and textual information [54, 2, 16], 3D-assisted recognition [26, 31, 37], introducing humans in the loop [40, 4, 7], and collecting larger amount of data [45, 47, 25, 17].

Intermediate representations in CNN Layer visualization [49] has shown that the intermediate layers of a CNN learn human-interpretable patterns from edges and corners to parts and objects. Regarding the discriminativeness of such patterns, there are two hypotheses. 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], a 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 re-

mains the same or slightly worse for the sake of better localization. On the other hand, [28, 21, 22] learn more discriminative representations by putting supervision on intermediate layers, usually by transforming the fully-connected layer output through another fully-connected layer followed by a loss layer. These transformations introduce a separation between the supervisory signal and internal filters that makes their methods difficult to visualize. A more recent related work is the popular SSD [32] detection framework; it associates a convolutional filter with either a particular category of certain aspect ratio or certain location coordinates. Compared to SSD, our architecture operates at a finer-level (small patches instead of objects) and is optimized for recognition.

3. Learning Discriminative Patch Detectors as a Bank of Convolutional Filters

We regard 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 $C \times H \times W$, each $C \times 1 \times 1$ vector across channels at fixed spatial location represents a small patch at a corresponding location in the original image. Suppose we have learned a 1×1 filter which has high response to a certain discriminative region; by convolving the feature map with this filter we obtain a heatmap. Therefore, a discriminative patch can be found simply by picking the location with the maximum value in the entire heatmap. This operation of spatially pooling the entire feature map into a single value is defined as Global Max Pooling (GMP) [56].

Two requirements are needed to make the feature map suitable for this idea. First, 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. Second, since fine-grained recognition involves accurate patch localization, the stride in the original image between adjacent patches should also be small. In early network architectures, the size and stride of the convolutional filters and pooling kernels were large. As a result, the receptive field of a single neuron in later convolutional layers was large, as was the stride between adjacent fields. Fortunately, the evolution of network architectures [36, 38, 15] has led to smaller filter sizes and pooling kernels. For example, in a 16-layer VGG network (VGG-16), the output of the 10^{th} convolutional layer `conv4_3` represents patches as small as 92×92 with stride 8, which is small and dense enough for our task given common CNN input size.

In the rest of Section 3, we demonstrate how a set of discriminative patch detectors can be 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. There are three key components in our design: an asymmetric two-stream structure to learn discriminative patches as well as global features (Section 3.1), convolutional filter supervision to ensure the discriminativeness of the patch detectors (Section 3.2) and non-random layer initialization to accelerate the network convergence (Section 3.3). We then extend our framework to handle patches of different scales (Section 3.4). We use VGG-16 for illustration, but our ideas are not limited to any specific network architecture as our experiments show.

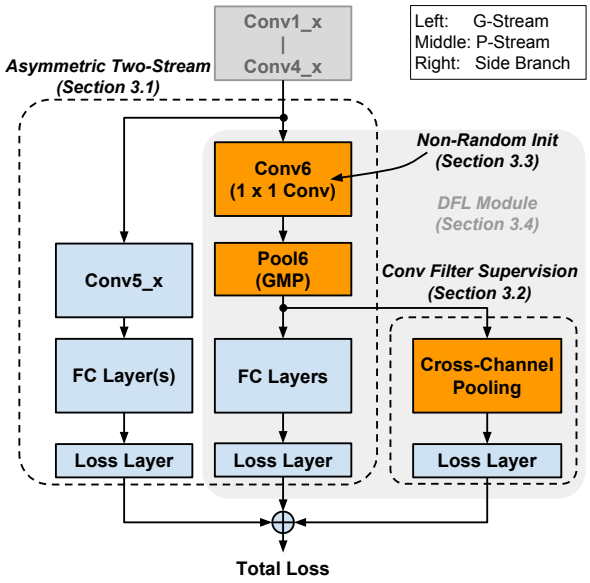

Figure 2. Overview of our framework, which consists of a) an asymmetric two-stream architecture to learn both the discriminative patches and global features, b) supervision imposed to learn discriminative patch detectors and c) non-random layer initialization. For simplicity, except GMP, all pooling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

3.1. 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. This component followed by a classifier (*e.g.*, fully-connected layers and a softmax layer) forms the discriminative patch stream (P-Stream) of our network, where the prediction is made by inspecting the responses of the discriminative patch detectors. 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.

The recognition of some fine-grained categories might also depend on global shape and appearance, so another stream preserves the further convolutional layers and fully connected layers, where the neurons in the first fully connected layer encode global information by linearly combining the whole convolutional feature maps. Since this stream

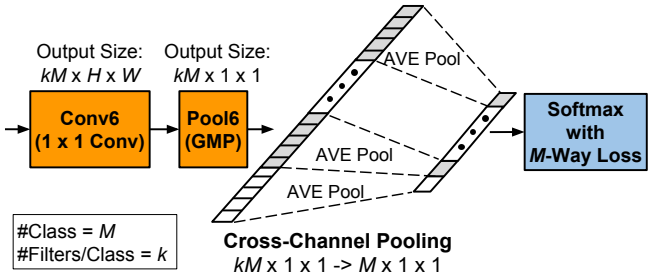

Figure 3. The illustration of our convolutional filter supervision. The filters in `conv6` are grouped into M groups, where M is the number of classes. The maximum responses in group i are averaged into a single score indicating the effect of the discriminative patches in Class i. The pooled vector is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

focuses on global features, we refer to it as the G-Stream.

We merge the two streams in the end.

3.2. Convolutional Filter Supervision

Using the network architecture described above, the 1×1 convolutional layer in the 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, shown in Figure 3 as part of the whole framework (the side branch) in Figure 2.

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 vector. Cross-Channel Pooling averages the values across every group of k dimensions as the response of a certain class, resulting in an M-dimensional vector. By feeding the pooled vector into an M-way 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. We use average instead of max pooling to encourage 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].

Since there is no learnable parameter between the softmax loss and the 1×1 convolutional layer, we directly adjust the filter weights via the loss function. In contrast, previous approaches which introduce intermediate supervision [28, 21, 22] have learnable weights (usually a fully-connected layer) between the side loss and the main network, which learn the weights of a classifier unused at test time. The main network is only affected by back-propogating the gradients of these weights. We believe this is a key difference of our approach from previous ones.

3.3. Layer Initialization

In practice, if the 1×1 convolutional layer is initialized randomly, with filter supervision it may converge to bad local minima. For example, the output vector of the Cross-Channel Pooling can approach all-zero or some constant to reduce the side loss during training, a degenerate solution. To overcome the issue, 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, using weak supervion without part annotations. Concretely, a patch is represented by a $C \times 1 \times 1$ vector at corresponding spatial location of the feature map. We extract the `conv4_3` features from the ImageNet pre-trained model and compute the energy at each spatial location (l_2 norm of each C-dimensional vector in a feature map). As shown in the first row of Figure 10, though not perfect, the heatmap of energy distribution acts as a reasonable indicator of useful patches. Then the vectors with high l_2 norms are selected via non-maximum suppression with small overlap threshold; k-means is performed over the selected C-dimensional vectors within Class i and the cluster centers are used as the initializations for filters from Class i. To increase their discriminativeness, we further whiten the initializations using [14] 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 show that the energy distribution 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 [29, 50, 18].

3.4. Extension: Multiple Scales

Putting Section 3.1 to 3.3 together, the resulting framework can utilize discriminative patches from a single scale. A natural and necessary extension is to utilize patches from multiple scales, since in visual domains such as birds and aircrafts, objects might have larger scale variations.

As discussed in Section 3.1, discriminative patch size depends on the receptive field of the input feature map. Therefore, multi-scale extension of our approach is equivalent to utilizing multiple feature maps. We regard the P-Stream and side branch (with non-random initialization) together as a "Discriminative Filter Learning" (DFL) module that is added after `conv4_3` in Figure 2. By simply adding the DFL modules after multiple convolutional layers we achieve multi-scale patch learning. In practice, feature maps produced by very early convolutional layers are not

suitable for class-specific operations since they carry information that is too low-level, therefore the DFL modules are added after several late convolutional layers in Section 4.

Our multi-layer branch-out is inspired by recent approaches in object detection [32, 46], where feature maps from multiple convolutional layers are directly used to detect objects of multiple scales. Compared with these works, our approach operates at a finer level and is optimized for recognition instead of localization.

4. Experiments

In the rest of this paper, we denote our approach by DFL-CNN, which is an abbreviation for *Discriminative Filter Learning within a CNN*. We use the following datasets:
CUB-200-2011 [39] has 11,788 images from 200 classes officially split into 5,994 training and 5,794 test images.
Stanford Cars [26] has 16,185 images from 196 classes officially split into 8,144 training and 8,041 test images.
FGVC-Aircraft [34] has 10,000 images from 100 classes officially split into 6,667 training and 3,333 test images.

4.1. Implementation Details

We first describe the basic settings of our DFL-CNN and then we introduce two higher-capacity settings. The input size of all our networks is 448×448, which is standard in the literature. We do not use part or bounding box (BBox) annotations and compare our method with other weakly-supervised approaches (without part annotation). In addition, no model ensemble is used in our experiments.

The network structure of our basic DFL-CNN is based on 16-layer VGGNet [36] and the DFL module is added after `conv4_3`, as illustrated exactly in Figure 2. In `conv6`, we set the number of filters per class to be 10. During Cross-Channel average pooling, the maximum responses of each group of 10 filters are pooled into one dimension. At initialization time, `conv6` is initialized in the way discussed in Section 3.3; other original VGG-16 layers are initialized from an ImageNet pretrained model directly (compared with "indirect" initialization of `conv6`) and other newly introduced layers are randomly initialized. After initialization, a single stage end-to-end training proceeds, with the G-Stream, P-Stream and side branch having their own softmax with cross-entropy losses with weights 1.0, 1.0 and 0.1 respectively. At test time, these softmax-with-loss layers are removed and the prediction is the weighted combination of the outputs of the three streams.

We extend DFL-CNN in two ways. The first extension, 2-scale DFL-CNN, was discussed in Section 3.4. In practice, two DFL modules are added after `conv4_3` and `conv5_2`, while the output of the last convolutional layer (`conv5_3`) is used by G-Stream to extract global information. The second extension shows that our approach applies to other network architectures, a 50-layer ResNet [15] in this case. Similar to VGGNet, ResNet also groups convolutional layers into five groups and our DFL module is added to the output of the fourth group (*i.e.* `conv4_x` in [15]). Initialization, training and testing of the two extended networks are the same as basic DFL-CNN.

4.2. Results

The results on CUB-200-2011, Stanford Cars and FGVC-Aircraft are displayed in Table 1, Table 2 and Table 3, respectively. In each table from top to bottom, the methods are separated into five groups, as discussed in Section 1, which are (1) fine-tuned baselines, (2) CNN features + multi-stage frameworks, (3) localization-classification subnets, (4) end-to-end feature encoding and (5) DFL-CNN. The basic DFL-CNN, 2-scale extension and ResNet extension in Section 4.1 are denoted by "DFL-CNN (1-scale) / VGG-16", "DFL-CNN (2-scale) / VGG-16" and "DFL-CNN (1-scale) / ResNet-50", respectively. Our VGG-16 based approach not only outperforms corresponding fine-tuned baseline by a large margin, but also achieves or outperforms state-of-the-art under the same base model; our best results further outperform state-of-the-art by a noticeable margin on all datasets, suggesting its effectiveness.

Earlier multi-stage frameworks built upon CNN features achieve comparable results, while they often require bounding box annotations and the multi-stage nature limits their potential. The end-to-end feature encoding methods have very high performance on birds, while their advantages diminish when dealing with rigid objects. The localization-classification subnets achieve high performance on various datasets, usually with a large number of network parameters. For instance, the STN [20] consists of an Inception localization network followed by four Inception classification networks without weight-sharing, and RA-CNN [9] consists of three independent VGGNets and two localization sub-networks. Our end-to-end approach achieves state-of-the-art with no extra annotation, enjoys consistent performance on both rigid and non-rigid objects, and has relatively compact network architecture.

Our approach can be applied to various network architectures. Most previous approaches in fine-grained recognition have based their network on VGGNets and previously reported ResNet-based results are less effective than VGG-based ones. Table 1, 2 and 3 shows that our ResNet baseline is already very strong, however our ResNet based DFL-CNN is able to outperform the strong baseline by a large margin (*e.g.* 3.3% absolute percentage on birds). This clearly indicates that CNN's mid-level learning capability can still be improved even though the network is very deep.

4.3. Ablation Studies

We conduct ablation studies to understand the components of our approach. These experiments use the basic

Method	Base Model	Accuracy (%)
FT VGGNet [9]	VGG-19	77.8
FT ResNet	ResNet-50	84.1
CoSeg(+BBox) [24]	VGG-19	82.6
PDFS [53]	VGGNet	84.5
STN [20]	Inception [19]	84.1
RA-CNN [9]	VGG-19	85.3
MA-CNN [55]	VGG-19	**86.5**
B-CNN [30]	VGG-16	84.1
Compact B-CNN [10]	VGG-16	84.0
Low-rank B-CNN [23]	VGG-16	84.2
Kernel-Activation [5]	VGG-16	85.3
Kernel-Pooling [8]	VGG-16	86.2
Kernel-Pooling [8]	ResNet-50	84.7
DFL-CNN (1-scale)	VGG-16	85.8
DFL-CNN (2-scale)	VGG-16	86.7
DFL-CNN (1-scale)	ResNet-50	**87.4**

Table 1. Comparison of our approach (DFL-CNN) to recent results on CUB-200-2011, **without** extra annotations (if not specified). For the finetuned (FT) baselines, we cite the best previously reported result if it is better than our implementation. The black-bold number represents the best previous result.

Method	Base Model	Accuracy (%)
FT VGGNet [9]	VGG-19	84.9
FT ResNet	ResNet-50	91.7
BoT(+BBox) [42]	VGG-16	92.5
CoSeg(+BBox) [24]	VGG-19	92.8
RA-CNN [9]	VGG-19	92.5
MA-CNN [55]	VGG-19	**92.8**
B-CNN [30]	VGG-16	91.3
Low-Rank B-CNN [23]	VGG-16	90.9
Kernel-Activation [5]	VGG-16	91.7
Kernel-Pooling [8]	VGG-16	92.4
Kernel-Pooling [8]	ResNet-50	91.1
DFL-CNN (1-scale)	VGG-16	93.3
DFL-CNN (2-scale)	VGG-16	**93.8**
DFL-CNN (1-scale)	ResNet-50	93.1

Table 2. Comparison of our approach (DFL-CNN) to recent results on Stanford Cars **without** extra annotations (if not specified).

Method	Base Model	Accuracy (%)
FT VGGNet	VGG-19	84.8
FT ResNet	ResNet-50	88.5
MGD(+BBox) [41]	VGG-19	86.6
BoT(+BBox) [42]	VGG-16	88.4
RA-CNN [9]	VGG-19	88.2
MA-CNN [55]	VGG-19	**89.9**
B-CNN [30]	VGG-16	84.1
Low-Rank B-CNN [23]	VGG-16	87.3
Kernel-Activation [5]	VGG-16	88.3
Kernel-Pooling [8]	VGG-16	86.9
Kernel-Pooling [8]	ResNet-50	85.7
DFL-CNN (1-scale)	VGG-16	91.1
DFL-CNN (2-scale)	VGG-16	**92.0**
DFL-CNN (1-scale)	ResNet-50	91.7

Table 3. Comparison of our approach (DFL-CNN) to recent results on FGVC-Aircraft **without** extra annotation (if not specified).

DFL-CNN framework and the CUB-200-2011 dataset.

Contribution of each stream Given a trained DFL-CNN, we investigate the contribution of each stream at test time. Table 4 shows that the performance of the G-Stream or P-Stream alone is mediocre, but the combination of the two is significantly better than either one alone, indicating that the global information and the discriminative patch information are highly complementary. Additionally, the side branch provides extra gain to reach the full performance in Table 1.

Settings	Accuracy (%)
G-Stream Only	80.3
P-Stream Only	82.0
G + P	84.9
G + P + Side	85.8

Table 4. Contribution of the streams *at test time* on CUB-200-2011. Note that at training time a *full* DFL-CNN model is trained, but the prediction only uses certain stream(s).

`pool6` Method	Accuracy (%)
GMP	85.8
GAP	80.4

Table 5. Effect of Global Max Pooling (GMP) vs. Global Average Pooling (GAP) on CUB-200-2011.

Layer Initialization	Filter Supervision	Accuracy (%)
-	-	82.2
✓	-	84.4
✓	✓	85.8

Table 6. Effect of intermediate supervision of DFL-CNN *at training time*, evaluated on CUB-200-2011.

Method	Without BBox (%)	With BBox (%)
FT VGG-16 [57]	74.5	79.8
DFL-CNN	85.8	85.7

Table 7. Effect of BBox evaluated on CUB-200-2011.

Effect of intermediate supervision We investigate the effect of Section 3.2 and 3.3 by training the DFL-CNN without certain component(s) and comparing with the full model. Table 6 shows a significant performance improvement when we gradually add the intermediate supervision components to improve the quality of learned discriminative filters. Note that Table 6 does not include "Filter Supervision without Layer Initialization" settings since it leads to failure to converge of P-Stream as mentioned in Section 3.3.

GMP vs. GAP More insight into the training process can be obtained by simply switching the pooling method of `pool6` in Figure 2. As can be seen from the Table 5, switching the pooling method from GMP to Global Average Pooling (GAP) leads to a significant performance drop such that the accuracy is close to "G-Stream Only" in Table 4. Therefore, although `conv6` is initialized to the same state, during training GMP makes the filters more discriminative by encouraging the 1×1 filters to have very high response at a certain location of the feature map and the gradients will only be back-propagated to that location, while GAP makes the P-Stream almost useless by encouraging the filters to have mediocre responses over the whole feature maps and the gradients affect every spatial location.

Unnecessary BBox. Since our approach, DFL-CNN, is able to utilize discriminative patches without localization, it

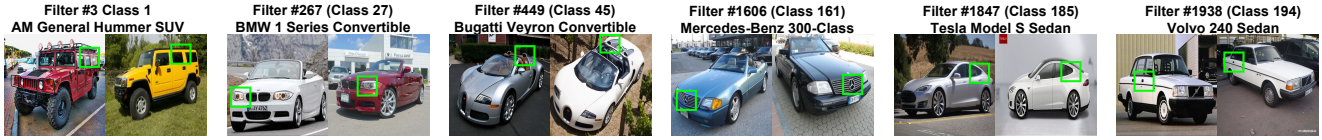

Figure 4. The visualization of top patches in Stanford Cars. We remap the spatial location of the highest activation in a feature map back to the patch in the original image. The results are highly consistent with human perception, and cover diverse regions such as head light (2^{nd} **column**), air intake (3^{th} **column**), frontal face (4^{th} **column**) and the black side stripe (**last column**).

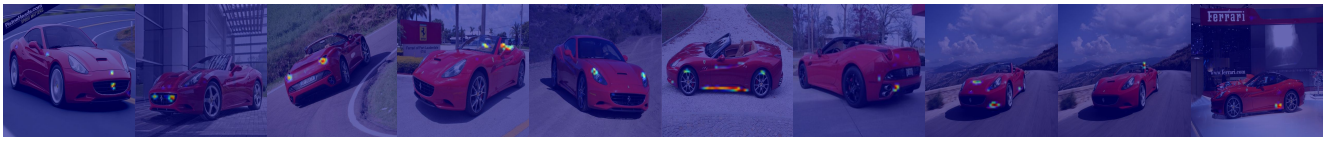

Figure 5. Sample visualization of all ten filter activations learned for one class (Class 102) by upsampling the `conv6` feature maps to image resolution, similar to [56]. The activations are disriminatively concentrated and cover diverse regions. Better viewed at 600%.

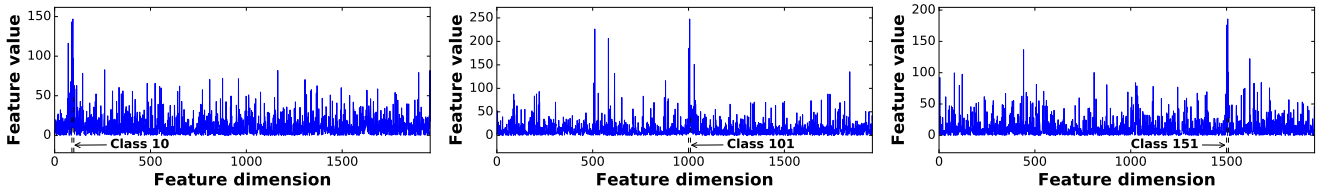

Figure 6. The `pool6` features averaged over all test samples from Class 10, 101 and 151 in Stanford Cars. The dash lines indicate the range of values given by the discriminative patch detectors belonging to the class. The representations peak at the corresponding class.

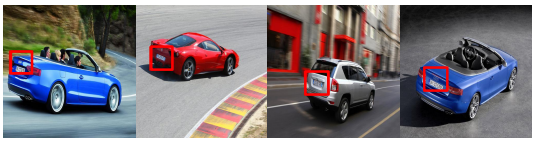

Figure 7. Visualization of a failure case, where the filter activates on commonly appeared licence plates.

is expected to be less sensitive to BBox than the fine-tuned baseline, as supported by the results in Table 7.

4.4. Visualization and Analysis

Insights into the behavior of our approach can be obtained by visualizing the effects of `conv6`, the 1×1 convolutional layer. To understand its behavior, we

- visualize patch activations. Since we regard each filter as a discriminative patch detector, we identify the learned patches by remapping spatial locations of top filter activations back to images. Figure 4 shows that 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 find 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` (VGG-16), through back propagation. By comparing the `conv4_3` features before and after training, we find that the spatial energy distributions of previous feature maps are changed in a discriminative fashion.

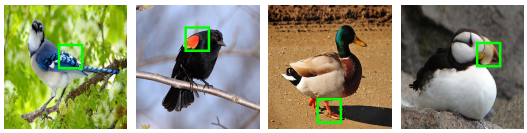

Figure 8. The visualization of patches in CUB-200-2011. We accurately localize discriminative patches without part annotations, such as the bright texture (**first image**), the color spot (**second image**), the webbing and beak (**third** and **forth** image).

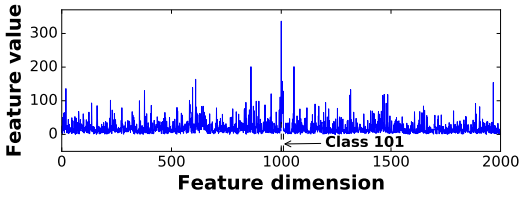

Figure 9. The averaged `pool6` features over all test samples from Class 101 in CUB-200-2011, peaky at corresponding dimensions.

4.4.1 Stanford Cars

The visualization of top patches found by some classes' 1×1 filters is displayed in Figure 4; the visualization of all ten filters learned for a sample class is displayed in Figure 5. 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 corresponding classes. Figure 4 shows that the top patches are very consistent with human perception. For instance, the 1847^{th} filter belonging to Class 185 (Tesla Model S) captures the distinctive tail of this type. Figure 5 shows that the filter activation are highly concentrated at

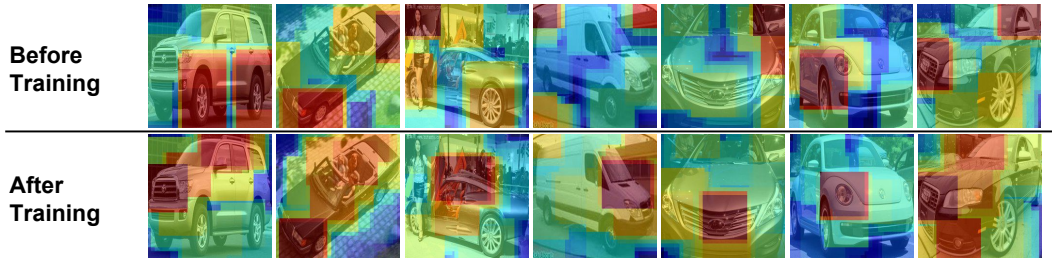

Figure 10. Visualization of the energy distribution of conv4_3 feature map before and after training for Stanford Cars. We remap each spatial location in the feature map back to the patch in the original image. After training in our approach, the energy distribution becomes more discriminative. For example, in the 1^{st} **column**, the high energy region shifts from the wheels to discriminative regions like the frontal face and the top of the vehicle; in the 2^{nd} **column**, after training the energy over the brick patterns is reduced; in the 3^{rd} **column**, the person no longer lies in high energy region after training; 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. More examples are interpretated in Section 4.4.1.

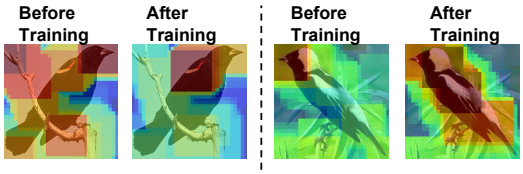

Figure 11. The energy distributions of conv4_3 feature maps before and after training in CUB-200-2011. After training, in the left example, the high energy region at the background branches is greatly shrinked and the energy is concentrated at the discriminative color spot; in the right example, more energy is distributed to the distinctive black-and-white wing and tail of the species.

discriminative regions and the ten filters cover diverse regions. The network can localize these subtle discriminative regions because: a) 1×1 filters correspond to small patch detectors in original image, b) the filter supervision, and c) the use of cluster centers as initialization promotes diversity.

The visualization of pool6 features is shown in Figure 6. 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 6 shows that our approach works as expected. As noticeable, the fine-grained similarity at patch-level (*e.g.* Audi A4 and Audi A6) and few common patterns (example shown in Figure 7) might explain the alternative peaks in Figure 6.

Most interesting is the effect of conv6 on the previous convolutional layer conv4_3 through back propagation. As discussed in Section 3.3, 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, as shown by Figure 10. 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 10, the features extracted from an ImageNet pre-trained 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 6^{th} column, the feature map has high energy initially at both the wheel and the head light; after training, the network has determined that a discriminative patch for that class (Volkswagen Beetle) is the head light rather than the wheels. Therefore, conv6 have beneficial effects on their previous layer during training.

4.4.2 CUB-200-2011

Figure 8 shows examples of the discriminative patches found by our approach. They include the texture and spots with bright color as well as specific shape of beak or webbing. Compared with visualizations of previous works not using part annotations (*e.g.* [24, 30]), our approach localizes such patches more accurately because our patch detectors operate over denser and smaller patches and do not have to be shared across categories.

Similar to cars, features from the next GMP layers are peaky at certain categories (Fig. 9). The energy distributions of previous convolutional features are also improved: high energy at background regions like branches is reduced and the discriminative regions become more focused or diverse according to different categories (Fig. 11).

5. Conclusion

We have presented an approach to fine-grained recognition based on learning a discriminative filter bank within a CNN framework in an end-to-end fashion without extra annotation. This is done via an asymmetric multi-stream network structure with convolutional layer supervision and non-random layer initialization. Our approach learns high-quality discriminative patches and obtains state-of-the-art performance on both rigid / non-rigid fine-grained datasets.

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. POOF: part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In *CVPR*, 2013. 2

[4] 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

[5] S. Cai, W. Zuo, and L. Zhang. Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization. In *ICCV*, 2017. 1, 2, 6

[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] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, 2017. 1, 2, 6

[9] J. Fu, H. Zheng, and T. Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, 2017. 1, 2, 5, 6

[10] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *CVPR*, 2016. 1, 2, 6

[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] 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

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

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 5

[16] X. He and Y. Peng. Fine-grained image classification via combining vision and language. In *CVPR*, 2017. 2

[17] S. Hou, Y. Feng, and Z. Wang. Vegfru: A domain-specific dataset for fine-grained visual categorization. In *ICCV*, 2017. 2

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

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 6

[20] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015. 1, 2, 5, 6

[21] Z. Jiang, Y. Wang, L. S. Davis, W. Andrews, and V. Rozgic. Learning discriminative features via label consistent neural network. In *WACV*, 2017. 3, 4

[22] X. Jin, Y. Chen, J. Dong, J. Feng, and S. Yan. Collaborative layer-wise discriminative learning in deep neural networks. In *ECCV*, 2016. 3, 4

[23] S. Kong and C. Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *CVPR*, 2017. 1, 2, 6

[24] J. Krause, H. Jin, J. Yang, and F. Li. Fine-grained recognition without part annotations. In *CVPR*, 2015. 1, 2, 6, 8

[25] 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

[26] 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, 2, 5

[27] M. Lam, B. Mahasseni, and S. Todorovic. Fine-grained recognition as hsnet search for informative image parts. In *CVPR*, 2017. 2

[28] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015. 3, 4

[29] 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

[30] T. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN models for fine-grained visual recognition. In *ICCV*, 2015. 1, 2, 6, 8

[31] 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

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016. 3, 5

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

[34] S. Maji, E. Rahtu, J. Kannala, M. B. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013. 1, 5

[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] J. Sochor, A. Herout, and J. Havel. Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition. In *CVPR*, 2016. 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. 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

[40] 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

[41] D. Wang, Z. Shen, J. Shao, W. Zhang, X. Xue, and Z. Zhang. Multiple granularity descriptors for fine-grained categorization. In *ICCV*, 2015. 1, 6

[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, 6

[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

[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] F. Yang, W. Choi, and Y. Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016. 5

[47] 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

[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, 8

[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

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

[53] 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, 6

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

[55] H. Zheng, J. Fu, T. Mei, and J. Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *ICCV*, 2017. 1, 6

[56] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016. 2, 3, 4, 7

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