Higher-order Integration of Hierarchical Convolutional Activations for Fine-grained Visual Categorization

Sijia Cai[1], Wangmeng Zuo[2], Lei Zhang[1]*
[1]Dept. of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[2]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
{csscai, cslzhang}@comp.polyu.edu.hk, cswmzuo@gmail.com

Abstract

The success of fine-grained visual categorization (FGVC) extremely relies on the modeling of appearance and interactions of various semantic parts. This makes FGVC very challenging because: (i) part annotation and detection require expert guidance and are very expensive; (ii) parts are of different sizes; and (iii) the part interactions are complex and of higher-order. To address these issues, we propose an end-to-end framework based on higher-order integration of hierarchical convolutional activations for FGVC. By treating the convolutional activations as local descriptors, hierarchical convolutional activations can serve as a representation of local parts from different scales. A polynomial kernel based predictor is proposed to capture higher-order statistics of convolutional activations for modeling part interaction. To model inter-layer part interactions, we extend polynomial predictor to integrate hierarchical activations via kernel fusion. Our work also provides a new perspective for combining convolutional activations from multiple layers. While hypercolumns simply concatenate maps from different layers, and holistically-nested network uses weighted fusion to combine side-outputs, our approach exploits higher-order intra-layer and inter-layer relations for better integration of hierarchical convolutional features. The proposed framework yields more discriminative representation and achieves competitive results on the widely used FGVC datasets.

1. Introduction

Deep convolutional neural networks (CNNs) have emerged as the new state-of-the-art for a wide range of visual recognition tasks. Nevertheless, it remains quite challenging to derive the effective discriminative representation for fine-grained visual categorization (FGVC), primarily due to subtle semantic differences between sub-ordinate categories. Conventional CNNs usually deploy the fully connected layers to learn global semantic representation and may not be suitable to FGVC. Therefore, leveraging local discriminative patterns in CNN is crucial to obtain more powerful representation, and recently has been intensively studied for FGVC.

Part-based representations [47, 3, 46, 34, 48] built on CNN features have been a predominant trend in FGVC. Such methods follow a detection module consisting of part detection and appearance modeling to extract regional features on deeper convolutional layers in R-CNN [12] based scenario. Then global appearance structure is incorporated to pool these regional features. Although these methods have yielded rich emporical returns, they still pose the following issues: (1) A considerable number of part-based methods [47, 3, 46] heavily rely on the detailed part annotations to train accurate part detectors, which is costly and further limits the scalability for large-scale datasets; moreover, identifying discriminative parts for specific fine-grained objects is quite challenging and often requires interaction with human or expert knowledge [4, 40]; (2) The discriminative semantic parts in images often appear at different scales. As each spatial unit in the deeper convolutional layer corresponds to a specific receptive field, activations from a single convolutional layer are limited in describing various parts with different sizes; (3) Exploiting the joint configuration of individual object parts is very important for object appearance modeling. A few works introduce additional geometric constraints for object parts including the popular deformable parts model [47], constellation model [34] and order-shape constraint [41]. One key disadvantage of these approaches is that they only characterize the first-order occurrences and relationships of very few parts, however, cannot be readily applied to model objects with more parts. Consequently, our focus is to capture the higher-order statistics of those semantic parts at different scales, and thus provide a more flexible way for global appearance modeling without the help of part annotation.

*This work is supported by HK GRC GRF grant (PolyU 152135/16E) and China NSFC grant (no. 61672446).

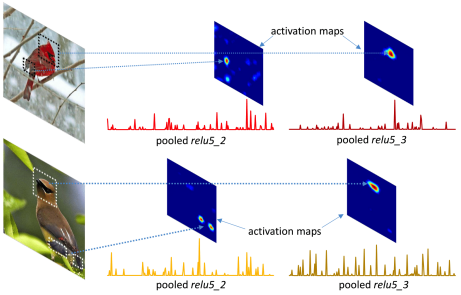

Figure 1. Visualization of several activation maps that corresponds to large responses of the sum-pooled vectors of two activation layers *relu5_2* and *relu5_3* in VGG-16 model.

In recent works [34, 48], the deeper convolutional filters are regarded as weak part detectors and the corresponding activations as the responses of detection, shown in Fig. 1. Motivated by this observation, instead of part annotations and explicit appearance modeling, we straightforwardly exploit the higher-order statistics from the convolutional activations. We first provide a perspective of matching kernel to understand the widely adopted mapping and pooling schemes on convolutional activations in conjunction with linear classifier. Linear mapping and direct pooling only capture the occurrence of parts. In order to capture the higher-order relations among parts, it is better to explore local non-linear matching kernels to characterize the higher-order part interactions (*e.g.*, co-occurrence). However, designing an appropriate CNN architechture that can be plugged with non-linear local kernels in an end-to-end manner is non-trivial. The kernel scheme is required to have explicit non-linear maps and be differentiable to facilitate back-propagation. One representative work is convolutional kernel network (CKN) [28], which provides a kernel approximation scheme to interpret CNNs. A related polynomial network [26] is to utilize polynomial activation functions as alternatives of ReLU in CNNs to learn non-linear interations of feature variables. Similarly, we leverage the polynomial kernel to serve in modeling higher-level part interactions and derive the polynomial modules that allow trainable structure built on CNNs.

With the kernel scheme, we extend our framework for higher-order integration of hierarchical convolutional activations. The effectiveness of fusing hierarchical features in CNNs has been widely reported in visual recognition. The benefits come from both the different discriminative capacities of multiple convolutional layers and the coarse-to-fine object description. However, the existing methods simply concatenate or sum multiple activations into a holistic representation [15], or adopt a decision level fusion to combine side-outputs from different layers [23, 42]. These methods, however, are limited in exploiting the intrinsic higher-order relationships of convolutional activations in either the intra-layer level or the inter-layer level. By using the kernel fusion on hierarchical convolutional activations, we can construct a richer image representation for cross-layer integration. Compared with the related works that perform feature fusion via learning multiple networks [8, 35, 24], our framework is easy to construct and more effective for FGVC.

2. Related work

2.1. Feature encoding in CNNs

Applying encoding techniques for the local convolutional activations in CNNs has shown significant improvements compared with the fully-connected outputs [7, 43]. In this case, the Vectors of Locally Aggregated Descriptors (VLAD) and Fisher Vectors (FV) as high-order statistics based representation can be readily applied. Gong *et al.* [13] propose to use VLAD to encode local features extracted from multiple regions of an image. In [9, 7, 43], the values of FV encoding on convolutional activations are discovered for scene, texture and video recognition tasks. However, regarding feature encoding as an isolated component is not the optimal choice for CNNs. Therefore, Lin *et al.* [24] propose a bilinear CNN (B-CNN) as codebook-free coding that allows end-to-end training for FGVC. The very recent work in [1] builds a weakly place recognition system by introducing a generalized VLAD layer that can be trained with off-the-shelf CNN models. An alternative for feature mapping is to exploit kernel approximation feature embedding. Yang *et al.* [45] introduce adaptive Fastfood transform in their deep fried convnets to replace the fully-connected layers, which is a generalization of the Fastfood transform for approximating kernels [22]. Gao *et al.* [11] implement an end-to-end structure to approximate degree-2 homogeneous polynomial kernel by utilizing random features and sketch techniques.

2.2. Feature fusion in CNNs

Compared with the fully connected layers capturing the global semantic information, convolutional layers preserve more instance-level details and exhibit diverse visual contents as well as different discriminative capacities, which are more meaningful to the fine-grained recognition task [2]. Recently a few works attempt to investigate the effectiveness of exploiting features from different convolutional layers [25, 44]. Long *et al.* [27] combine the feature maps from intermediate level and high level convolutional layers in their fully convolutional network to provide both finer details and higher-level semantics for better image segmentation. Hariharan *et al.* [15] introduce hypercolumns for localization and segmentation, where convolutional activations at a pixel of different feature maps are concatenated as a vector as a pixel descriptor. Similarly, Xie and Tu [42] present a holistically-nested edge detection scheme in which the sideoutputs are added after several lower convo-

lutional layers to provide deep supervision for predicting edges at multiple scales.

3. Kernelized convolutional activations

Most part-based CNN methods for FGVC consist of two components: (i) feature extraction for semantic parts on the last convolutional layer, and (ii) spatial configuration modeling for those parts to produce discriminative image representation. In this work, we treat the convolutional filter as part detector, and then the convolutional activations in a single spatial position can be considered as the part descriptions. Therefore, instead of explicit part extraction, we introduce polynomial predictor to integrate a family of local matching kernels for modeling higher-order part interactions and derive powerful representation for FGVC.

3.1. Matching kernel and polynomial predictor

Suppose that an image I is passed by a plain CNN, and we denote the 3D activations $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{K\times M\times N}$ extracted from some specific convolutional layer as a set of K-dimensional descriptors $\{\boldsymbol{x}_p\}_{p\in\Omega}$, where K is the number of feature channels, $\boldsymbol{x}_p$ represents the descriptor at a particular position p over the set Ω of valid spatial locations ($|\Omega| = M \times N$). We first consider the matching scheme $\mathcal{K}$ for activation sets $\boldsymbol{\mathcal{X}}$ and $\bar{\boldsymbol{\mathcal{X}}}$ from two images, in which the set similarity is measured via aggregating all the pairwise similarities among the local descriptors:

$$\mathcal{K}(\boldsymbol{\mathcal{X}}, \bar{\boldsymbol{\mathcal{X}}}) = Agg(\{k(\boldsymbol{x}_p, \bar{\boldsymbol{x}}_{\bar{p}})\}_{p\in\Omega,\bar{p}\in\bar{\Omega}}) = \psi(\boldsymbol{\mathcal{X}})^T\psi(\bar{\boldsymbol{\mathcal{X}}}), \quad (1)$$

where $k(\cdot)$ is some kernel function between individual descriptors of two activation sets, $Agg(\cdot)$ is some set-based aggregation function, $\psi(\boldsymbol{\mathcal{X}})$ and $\psi(\bar{\boldsymbol{\mathcal{X}}})$ are the vector represetations for sets. It is worth noting that the construction of $\mathcal{K}$ presented above is decomposed into two steps in CNNs: feature mapping and feature aggregation. The mapping step maps each local descriptor $\boldsymbol{x} \in \mathbb{R}^K$ as $\phi(\boldsymbol{x}) \in \mathbb{R}^D$ in elaborated feature space. The aggregating step produces an image-level representation $\psi(\boldsymbol{\mathcal{X}})$ from the set $\{\phi(\boldsymbol{x}_p)\}_{p\in\Omega}$ through some pooling function $g(\cdot)$.

The key for FGVC is to discover and represent those local regions which share common appearances within the same category while exhibiting distinctive difference across categories. Based on the matching scheme $\mathcal{K}$ in Eqn. (1), appropriate pooling operators have been designed to efficiently prune non-discriminative matching subset while retaining those highly discriminative ones into image representation. Among them, sum pooling assigns equal weights to each position, and does not emphasize any position. Max pooling only considers the most significant position, which results in enormous information loss and is prone to small interference. Other pooling operators such as generalized max pooling [31] and ℓ_p-norm pooling [10] may be effective in discovering informative regions, but the feasible end-to-end schemes are unclear. Our attention is to model the higher-order relationships for discriminative representation of local patch and design suitable local mapping function ϕ which can be stacked upon CNN for end-to-end training. Thus, we simply adopt $g(\cdot)$ as the global sum pooling, in which case we denote it as:

$$\psi(\boldsymbol{\mathcal{X}}) = g(\{\phi(\boldsymbol{x}_p)\}_{p\in\Omega}) = \sum_{p\in\Omega}\phi(\boldsymbol{x}_p). \quad (2)$$

The above matching underpinning highlights the advantage of generating image-level representation compatible with linear predictors, which can be interpreted as the linear combination of all local compositions accordingly:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x})\rangle, \quad (3)$$

where $\boldsymbol{w}$ is the parameter of predictor, we omit the bias term and position subscript p here for later convenience. As our aim is to capture more complex and higher-order relationships among parts, to this end, we propose the following polynomial predictor:

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} w_k x_k + \sum_{r=2}^{R}\sum_{k_1,\ldots,k_r} \mathcal{W}^r_{k_1,\ldots,k_r}(\prod_{s=1}^{r} x_{k_s}), \quad (4)$$

where R is the maximal degree of part interactions, $\boldsymbol{\mathcal{W}}^r$ is a r-order tensor which contains the weights of degree-r variable combinations in $\boldsymbol{x}$. For instance, when $r = 3$, $\mathcal{W}_{i,j,k}$ is the weight of $x_i x_j x_k$. We discuss different polynomial predictors as well as their corresponding kernels as follows:

1) **Linear kernel:** $k(\boldsymbol{x}, \bar{\boldsymbol{x}}) = \langle \boldsymbol{x}, \bar{\boldsymbol{x}}\rangle$ is the most simple kernel that refers to an identity map $\phi : \boldsymbol{x} \mapsto \boldsymbol{x}$, which is identical to the polynomial predictor of degree-1: $f(\boldsymbol{x}) = \sum_{k=1}^{K} w_k x_k$.

2) **Homogeneous polynomial kernel:** $k(\boldsymbol{x}, \bar{\boldsymbol{x}}) = \langle \boldsymbol{x}, \bar{\boldsymbol{x}}\rangle^r$ has shown the superiority in characterizing the intrinsic manifold structure of dense local descriptors [5]. The induced non-linear map $\phi : \boldsymbol{x} \mapsto \otimes_r \boldsymbol{x}$, where $\otimes_r \boldsymbol{x}$ is a tensor defined by the r-order self-outer product [32] of $\boldsymbol{x}$, is able to model all the degree-r interactions between variables. Its polynomial predictor obeys the following form:

$$f(\boldsymbol{x}) = \sum_{k_1,\ldots,k_r} \mathcal{W}^r_{k_1,\ldots,k_r}(\prod_{s=1}^{r} x_{k_s}). \quad (5)$$

Notice that the polynomial predictor of degree-2 homogeneous polynomial kernel is defined as $\sum_{i,j} W_{i,j} x_i x_j$, which captures all pairwise/second-order interactions between variables and is an increasingly popular model in classification tasks [24].

3) **Positive definite kernel:** as discussed in [18], the positive definite kernel $k(\boldsymbol{x}, \bar{\boldsymbol{x}}) : (\boldsymbol{x}, \bar{\boldsymbol{x}}) \mapsto f(\langle \boldsymbol{x}, \bar{\boldsymbol{x}}\rangle)$ defines an analytic function which admits a Maclaurin expansion with only nonnegative coefficients, *i.e.*, $f(x) =$

$\sum_{r=0}^{\infty} a_r x^r$, $a_r \geq 0$. For instance, a non-homogeneous degree-2 polynomial kernel $(\langle \boldsymbol{x}, \bar{\boldsymbol{x}} \rangle + 1)^2$ corresponds to a polynomial predictor that captures all single and pairwise interactions between variables. It also indicates that the positive definite kernel can be arbitrarily accurate approximation of polynomial kernels in principle of sufficiently high degree polynomial expansions for target functions.

3.2. Tensor learning for polynomial kernels

Before deriving the end-to-end CNN architecture for learning the parameters in Eqn. (4), we first reformulate the polynomial predictor into a more concise tensor form:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{r=2}^{R} \langle \boldsymbol{\mathcal{W}}^{\boldsymbol{r}}, \otimes_{\boldsymbol{r}} \boldsymbol{x} \rangle, \tag{6}$$

where $\langle \boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{V}} \rangle$ is the inner product of two same-sized tensors $\boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{V}} \in \mathbb{R}^{K_1 \times \cdots \times K_r}$, which is defined as the sum of the products of their entries. It is observed that the tensor $\otimes_r \boldsymbol{x}$ comprises all the degree-r monomials in $\boldsymbol{x}$. Therefore, any degree-r homogeneous polynomial predictor satisfies $\langle \boldsymbol{\mathcal{W}}^r, \otimes_r \boldsymbol{x} \rangle$ for some r-order tensor $\boldsymbol{\mathcal{W}}^r$; likewise, any r-order tensor $\boldsymbol{\mathcal{W}}^r$ determines a degree-r homogenous polynomial predictor. This equivalence between polynomials and tensors motivates us to transform the parameter learning of polynomial predictor into tensor learning.

Rather than estimating the variable interations in tensors independently, an alternative method is tensor decomposition [19] which breaks the independence of interaction parameters and estimates the reliable interaction parameters under high sparsity. Tensor decomposition is widely used in tensor machines [38] for sparse data based regression, which circumvents the parameter storage issue and achieves better generalization in practice. We then embrace the rank-one tensor decomposition [19] in our next step of tensor learning for consideration of two aspects: the high sparsity of activations in deeper layers of CNNs and the parameter sharing of convolutional filters.

We first briefly review the notations and definitions in the area of rank-one tensor decomposition: the outer product of vectors $\boldsymbol{u}_1 \in \mathbb{R}^{K_1}, \ldots, \boldsymbol{u}_r \in \mathbb{R}^{K_r}$ is the $K_1 \times \cdots \times K_r$ rank-one tensor that satisfies $(\boldsymbol{u}_1 \otimes \cdots \otimes \boldsymbol{u}_r)_{k_1 \ldots, k_r} = (\boldsymbol{u}_1)_{k_1} \cdots (\boldsymbol{u}_r)_{k_r}$. The rank-one decomposition for a tensor $\boldsymbol{\mathcal{W}}$ is defined as $\boldsymbol{\mathcal{W}} = \sum_{d=1}^{D} \alpha^d \boldsymbol{u}_1^d \otimes \cdots \otimes \boldsymbol{u}_r^d$, where α^d is the weight for d-th rank-one tensor, D is the rank of the tensor if D is minimal. We then apply the rank-one approximation [19] for each r-order tensor $\boldsymbol{\mathcal{W}}^r$ and present the following alternative form of polynomial predictor:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{r=2}^{R} \langle \sum_{d=1}^{D^r} \alpha^{r,d} \boldsymbol{u}_1^{r,d} \otimes \cdots \otimes \boldsymbol{u}_r^{r,d}, \otimes_r \boldsymbol{x} \rangle. \tag{7}$$

In order to learn $\boldsymbol{w}$, $\alpha^{r,d}$ and $\boldsymbol{u}_s^{r,d}$ ($r = 2, \ldots, R, s = 1, \ldots, r, d = 1, \ldots, D^r$), in next section, we show that all the parameters can be absorbed into the conventional trainable modules in CNNs.

3.3. Trainable polynomial modules

According to the tensor algebra, the Eqn. (7) can be further rewritten as:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{r=2}^{R} \sum_{d=1}^{D^r} \alpha^{r,d} \prod_{s=1}^{r} \langle \boldsymbol{u}_s^{r,d}, \boldsymbol{x} \rangle \tag{8}$$

$$= \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{r=2}^{R} \langle \boldsymbol{\alpha}^r, \boldsymbol{z}^r \rangle \tag{9}$$

where the d-th element of the vector $\boldsymbol{z}^r \in \mathbb{R}^{D_r}$ is $\prod_{s=1}^{r} \langle \boldsymbol{u}_s^{r,d}, \boldsymbol{x} \rangle$ which characterizes the degree-r variable interactions under a single rank-one tensor basis. $\boldsymbol{\alpha}^r = [\alpha^{r,1}, \ldots, \alpha^{r,D^r}]^T$ is the associated weight vector of all D^r rank-one tensors. A key observation of Eqns. (8) (9) is that we are able to decouple the parameters into $\{\boldsymbol{w}, \boldsymbol{\alpha}^2, \ldots, \boldsymbol{\alpha}^R\}$ and $\{\{\boldsymbol{u}_s^{r,d}\}_{s=1,\ldots,r;d=1,\ldots D_r}\}_{r=2,\ldots,R}$. Notice that for each s, we can first deploy $\{\boldsymbol{u}_s^{r,d}\}_{d=1,\ldots D_r}$ as a set of D_r 1×1 convolutional filters on $\boldsymbol{\mathcal{X}}$ to generate a set of feature maps $\boldsymbol{\mathcal{Z}}_s^r$ of dimension $D^r \times M \times N$. Then, the feature maps $\{\boldsymbol{\mathcal{Z}}_s^r\}_{s=1,\ldots,r}$ from different ss are combined by element-wise product to obtain $\boldsymbol{\mathcal{Z}}^r = \boldsymbol{\mathcal{Z}}_1^r \odot \cdots \odot \boldsymbol{\mathcal{Z}}_r^r$. Therefore, $\{\boldsymbol{u}_s^{r,d}\}_{s=1,\ldots,r;d=1,\ldots D_r}$ can be treated as a polynomial module in learning degree-r polynomial features. As for the former parameter group, it can be easily embedded into the learning of the classifier for the concatenated polynomial features. Refering to Eqn. (8), the derivatives for $\boldsymbol{x}$ and each degree-r convolutional filter $\boldsymbol{u}_s^{r,d}$ in back propagation process can be achieved by:

$$\frac{\partial \ell}{\partial \boldsymbol{x}} = \frac{\partial \ell}{\partial \boldsymbol{y}^r} \sum_{d=1}^{D^r} \sum_{s=1}^{r} (\prod_{t \neq s} \langle \boldsymbol{u}_t^{r,d}, \boldsymbol{x} \rangle) \boldsymbol{u}_s^{r,d} \tag{10}$$

$$\frac{\partial \ell}{\partial \boldsymbol{u}_s^{r,d}} = \frac{\partial \ell}{\partial \boldsymbol{y}^r} (\prod_{t \neq s} \langle \boldsymbol{u}_t^{r,d}, \boldsymbol{x} \rangle) \boldsymbol{x} \tag{11}$$

where $\boldsymbol{y}^r = g(\boldsymbol{\mathcal{Z}}^r) = g(\{\boldsymbol{z}^r\})$ is the pooled feature representation for degree-r polynomial module, ℓ is the loss associated with $\boldsymbol{y}^r$. On this basis, we can embrace those polynomial modules with the trainable CNN architectures and are able to model the higher-order part statistics of any degree. Even though the dominant level of those highly-correlated parts will be enhanced with a larger r, the high-order tensor usually needs large D^r to guarantee a good approximation. Therefore, a relative small degree r should be considered in practice because a high-degree polynomial module increases the computational cost in back propagation, *i.e.*, Eqns. (10) (11), and the induced high dimensionality of feature would cause over-fitting.

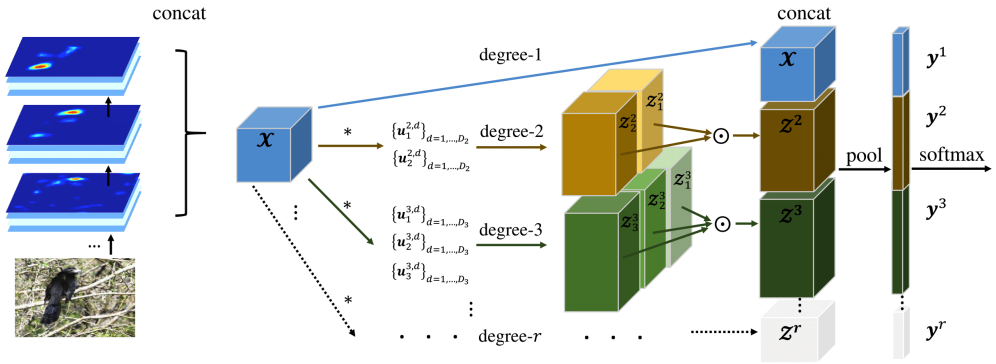

Figure 2. Illustration of our integration framework. The convolutional activation maps are concatenated as $\boldsymbol{\mathcal{X}} = \text{concat}(\boldsymbol{\mathcal{X}}^1, \ldots, \boldsymbol{\mathcal{X}}^L)$ and fed into different branches. For r-th branch ($r \geq 2$), the degree-r polynomial module consisting of r groups of 1×1 convolutional filters is deployed to obtain r sets of feature maps $\{\boldsymbol{\mathcal{Z}}_s^r\}_{s=1,\ldots,r}$. Then $\{\boldsymbol{\mathcal{Z}}_s^r\}_{s=1,\ldots,r}$ are integrated as $\boldsymbol{\mathcal{Z}}^r$ by applying element-wise product $\odot$. At last, $\boldsymbol{\mathcal{X}}$ and all $\boldsymbol{\mathcal{Z}}^r$s are concatenated as the degree-r polynomial features, following by sum pooling layer to obtain the pooled representation $\boldsymbol{y} = \text{concat}(\boldsymbol{y}^1, \ldots, \boldsymbol{y}^L)$ with the dimension of $\sum_{r=1}^{R} D_r$ (D_1 denotes the channel number of $\boldsymbol{\mathcal{X}}$), and softmax layer.

4. Hierarchical convolutional activations

4.1. Higher-order integration using kernel fusion

The polynomial predictor provides a good measure for the highly-correlated parts but the activations on individual convolutional layer are not sufficient to describe the part relations from different levels of abstraction and scale. Consequently, we investigate a kernel fusion scheme to combine the hierarchical convolutional activations. Suppose that the local activation descriptor sets from L convolutional layers at spatial correspondences for two images are denoted as $\boldsymbol{\psi}_I : \{\boldsymbol{x}^l\}_{l=1}^L$ and $\boldsymbol{\psi}_{\bar{I}} : \{\bar{\boldsymbol{x}}^l\}_{l=1}^L$. Then we generalize ϕ under linear factorization to fuse the local activations from multiple convolutional layers as below:

$$\begin{aligned} k(\boldsymbol{\psi}_{\boldsymbol{I}}, \boldsymbol{\psi}_{\bar{\boldsymbol{I}}}) &= \langle \phi(\{\boldsymbol{x}^l\}_{l=1}^L), \phi(\{\bar{\boldsymbol{x}}^l\}_{l=1}^L) \rangle \\ &= \sum_{l=1}^{L} \eta_l \langle \phi^l(\boldsymbol{x}^l), \phi^l(\bar{\boldsymbol{x}}^l) \rangle, \end{aligned} \tag{12}$$

where η_l is the weight for the matching scores in l-th layer. The above kernel fusion can be re-interpreted as performing polynomial feature extraction at each layer and fusing them in latter phase. Recently, hypercolumn [15] suggests a simple feature concatenation manner to combine different feature maps in CNNs for pixel-level classification, which motivates us to adopt the similar way in our polynomial kernel fusion. Thereby, we assume a holistic mapping ϕ for all layers, *i.e.*, $\sum_{l=1}^{L} \sqrt{\eta_l} \phi^l(\boldsymbol{x}^l) \rightarrow \phi(\text{concat}(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^L))$ with weights $\sqrt{\eta_l}$s be merged into element-wise scale layers. It should be noted that the spatial resolutions of different convolutional layers need to be consistent for concatenation operation. Alternatively, we can add pooling layers or spatial resampling layers to meet this requirement. In this sense, the expansion of ϕ by Eqn. (4) yields two groups of variable interactions: $\prod_{k_l} x_{k_l}^l$ that characterizes the interactions of parts in the l-th layer; and $\prod_{k_l, k_q} x_{k_l}^l x_{k_q}^q$ (where $l \neq q$) that captures additional information of multi-scale part relations from the l-th layer and q-th layer.

4.2. Integration architecture for deeper layers

Although the kernel fusion scheme enables polynomial predictor for integrating hierarchical convolutional activations, it may not perform and scale well in case where large numbers of layers involoved. We argue that only the convolutional activations from very deep layers refer to the responses of discriminative semantic parts. That is consistent with the recent studies [34, 48] which regard the convolutional filters in deeper layers as weak part detectors. In our experiments, we demonstrate that the integration of the last three convolutional activation layers (*i.e.*, *relu5_1*, *relu5_2*, and *relu5_3* in VGG-16 model [36]) is fairly effective to obtain satisfactory performance. Even though more lower layers could be involved, the effect is less obvious on the improvement but higher computational complexity on both training and testing phases. Fig. 2 presents our CNN architecture for integrating multiple convolutional layers. Compared with the B-CNN methods [24, 11] focusing only on the degree-2 part statistics, our approach provides a general solution to model complex part interactions from hierarchical layers in differnt degrees and its superiority will be demonstrated in experiments.

5. Experiments

In this section, we evaluate the effectiveness of our proposed integration framework on three fine-grained categorization datasets: Caltech-UCSD Bird-200-2011 (CUB) [39], Aircraft [29] and Cars [21]. The experimental comparisons with state-of-the-art methods indicate that effective feature integration from CNN is a promising solution for FGVC in contrast with the requirements of massive ex-

ternal data or detailed part annotation.

5.1. Datasets and Implementation Details

CUB dataset contains 11,788 bird images. There are altogether 200 bird species and the number of images per class is about 60. The significant variations in pose, viewpoint and illumination inside each class make this dataset very challenging. We adopt the publicly available split [39], which use nearly half of the dataset for training and the other half for testing.

Aircraft dataset has 100 different aircraft model variants, giving 100 images for each model. The aircrafts appear at different scales, design structures and appearances. We adopt the training/testing split protocol provideded by [29] to perform our experiments.

Cars dataset consists of 16,185 images from 196 car classes. Each class has about 80 images with different car sizes and heavy clutter background. We use the same split provided by [21], divided with 8,144 images for training and 8,041 images for testing.

Implementation details: our networks on all datasets are fine-tuned on the VGG-16 model pretrained on ILSVRC-2012 dataset [33] for fair comparison with most state-of-the-art FGVC methods. The framework can be also applied to the recently proposed network architectures such as Inception [37] and ResNet [16]. We remove the last three fully-connnected layers and construct a directed acyclic graph (DAG) to combine all the components in our framework. Before fed into softmax layer, we first pass pooled polynomial features through ℓ_2 normalization step. We then use logistic regression to intialize the parameters of classification layer, and adopt Rademacher vectors (*i.e.*, each of its components are chosen independently using a fair coin toss from the set $\{-1, 1\}$) as good initializations [18] of homogenous polynomial kernels for the 1×1 convolutional filters. In training phase, following [24], we transform the input image by cropping the largest image region around its center, resizing it to 448×448, and creating its mirrored version to double the training set. During fine-tuning, the learning rates of those pre-trained VGG-16 layers and the newly added layers, including 1×1 convolutional layers and classification layer, are initialized as 0.001. We train all the networks using stochastic gradient descent with a batch size of 16, momentum of 0.9. In testing phase, we follow the popular CNN-SVM scheme [24], *i.e.*, use softmax loss in training and then perform evaluation on the extracted features by SVM. Our code is implemented on the open source MatConvNet framework with a single NVIDIA GeForce GTX TITAN X GPU and can be downloaded at `http://www4.comp.polyu.edu.hk/~cslzhang/code/hihca.zip`.

5.2. Analysis of the proposed framework

5.2.1 Effect of number of 1×1 convolutional filters

To validate the effectiveness of introducing tensor decomposition in our polynomial predictor, we investigate the effect of different D^r for the approximation of each r-order tensor $\mathcal{W}^r$. We first evaluate the classification accuracies on the CUB dataset on a single layer *relu5_3* using different homogeneous polynomial kernels for solely modeling the degree-r variable interactions, *i.e.*, $x_i, x_i x_j, x_i x_j x_k, x_i x_j x_k x_l$. The number D^r for degree-r convolutional filters varies from 512 to 32,768. The results are shown in Fig. 3. As expected, increasing D^r leads higher accuracies on all degrees. Interestingly, when D^r is small, degree-2 always leads a higher accuracy than those with higher degrees, which indicates that modeling higher-order part interactions often yields a tensor of dense parameters. It is observed that the performance gain is slight when the number D^r increases from 8,192 to 32,768, which infers that a relative sparse tensor $\mathcal{W}^r$ can comprehensively encode the distinguishing part interactions of fine-grained objects from the very sparse activation features. Therefore, we uniformly use 8,192 1×1 convolutional filters in all the polynomial modules in consideration of feature dimension, computational complexity as well as accuracy.

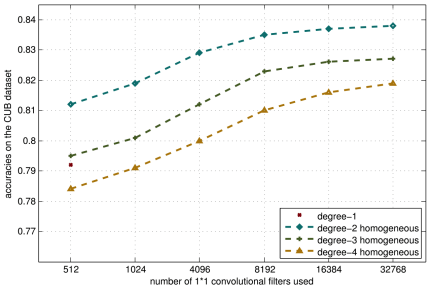

Figure 3. Accuracies achieved by using polynomial kernels with varied numbers of 1×1 convolutional filters on the CUB dataset.

5.2.2 Effect of polynomial degree r

We further demonstrate the superiority of using higher-order part interactions both with and without finetuning on the CUB dataset in Table 1. We observe that the degree-2 polynomial kernel significantly outperforms the linear kernel. It implies that the co-occurrence statistics is very effective in capturing part relations, which is more informative in distinguishing objects with homogeneous appearance than the simple part occurrence statistics. The accuracy degrades considerably as the degree r increases from 2 to 6, which might be explained by the fact the low-degree interactions with high counts are more reliable. As the reliable high-degree interactions are usually a few in number, the sum

pooling will abate those scarce interactions in the pooled polynomial representation, which weakens the discriminative ability of the final concatenated representation. Table. 2 lists the frame-per-second (FPS) comparison in both training and testing phases using different polynomial kernels. Since there is high computational overhead involved in the polynomial modules in the network, a large degree r will significantly slow the speed. Therefore, we suggest to adopt 2 as the practical degree in all the experiments in Section 5.3 even though degree-3 kernel can achieve slightly better results on Aircraft and Cars datasets.

Table 1. Accuracy comparison with different non-homogeneous polynominal kernels.

r	1	2	3	4	5	6
non-ft	75.7	**78.3**	76.4	74.6	72.4	71.2
ft	79.2	**83.7**	83.3	82.0	81.1	79.5

Table 2. FPS with different non-homogeneous polynomial kernels.

r	2	3	4	5	6
Training	9.7	7.4	5.5	4.2	2.8
Testing	29.8	23.7	18.3	14.5	10.4

5.2.3 Effect of feature integration

We then provide details of the results by using higher-order integration for hierarchical convolutional activations. We focus on *relu5_1*, *relu5_2* and *relu5_3* as they exhibit good capacity in capturing semantic part information compared with lower layers. And we analyze the impact factors of layers, kernels, and finetuning on the CUB dataset. The accuracies are obtained under five polynomial kernels including linear kernel, degree-2 homogeneous kernel, degree-2 non-homogeneous (single + pairwise interactions), degree-3 homogeneous kernel and degree-3 non-homogeneous kernel (single + pairwise + triple interactions). We consider the following baselines: *relu5_3* uses only *relu5_3* activations. *relu5_3+relu5_2*, *relu5_3+relu5_1* and *relu5_2+relu5_1* are integration baselines that use 2 layers. *relu5_1+relu5_2+relu5_3* is the full integration of three layers. The results in Table 3 demonstrate that the performance gain of our framework comes from three factors: (i) higher-order integration, (ii) finetuning, (iii) multiple layers. Notably, we observe the remarkable performance benefits on the baseline *relu5_3+relu5_2* and the full model of three layers by exploiting the degree-2 and degree-3 polynomial kernels, which implies that the discriminative power can be enhanced by the complementary capacities of hierarchical convolutional layers compared with the isolated *relu5_3* layer. As the baseline *relu5_3+relu5_2* already presents the best performance, thus we set the feature integration as *relu5_3+relu5_2* in all the experiments in Section 5.3.

Table 3. Accuracy comparison with different baselines.

	r5_3	*r5_3+ r5_2*	*r5_3+ r5_1*	*r5_2+ r5_1*	*r5_3+ r5_2+ r5_1*
degree-1					
non-ft	75.7	**77.2**	75.5	68.9	77.0
ft	79.2	80.4	79.3	71.1	**80.8**
degree-2 homogeneous					
non-ft	77.2	78.1	77.5	72.3	**78.4**
ft	83.5	**85.0**	83.3	76.0	84.9
degree-2 non-homogeneous					
non-ft	78.3	78.5	77.5	72.1	**78.6**
ft	83.7	**85.3**	83.6	76.5	85.1
degree-3 homogeneous					
non-ft	75.7	**76.9**	76.0	70.7	76.1
ft	82.3	**83.8**	81.5	74.1	83.3
degree-3 non-homogeneous					
non-ft	76.4	**78.2**	77.4	72.3	78.1
ft	83.3	**84.6**	82.1	75.4	84.5

We also compare our higher-order integration with hypercolumn [15] and HED [42] based feature integrations. Since the original hypercolumn and HED are introduced for pixel-wise classification, for fair comparison, we revise hypercolumn as the feature concatenation of *relu5_3*, *relu5_2* and *relu5_1*, following by max pooling (denoted as Hypercolumn*); and revise HED by training classifiers for the pooled activation features at each layer and then fuse the predictions (denoted as HED*). Table 4 shows that our integration framework is significantly superior to Hypercolumn* and HED*. This is not surprising since Hypercolumn* and HED* can be treated as degree-1 integration to some extent.

Table 4. Accuracy comparison with different feature integrations.

Degree-2 integration	Hypercolumn*	HED*
85.1	80.9	82.3

5.3. Comparison with state-of-the-art methods

5.3.1 Results on the CUB dataset

We first compare our framework along with both the annotation-based methods (*i.e.*, using object bounding boxes or part annotations) and annotation-free methods (*i.e.*, only using image-level labels) on the CUB dataset. As shown in Table 5, unlike the state-of-the-art result obtained from SPDA-CNN (85.1%) [46] which relys on the additional annotations of seven parts, we can still achieve a comparable accuracy of 85.3% with only image-level labels and significant improvements over PB R-CNN [47] and FG-Without [20]. Furthermore, our method is slightly inferior to BoostCNN [30] and outperforms all other annotation-free methods with a modest improvement (about 1%) compared with STN [17], B-CNN [24] and PDFS [48]. However, STN [17] uses a better baseline CNN (Inception [37]) than our VGG-16 network and PDFS [48] cannot be trained

by end-to-end manner. B-CNN [24] attempts to achieve the feature complementary based on the outer product of convolutional activations from two networks (i.e., VGG-M and VGG-16). However, our framework shows that the better complementarity can be achieved by exploiting the natural hierarchical structures of CNNs. BoostCNN uses BCNN as the base CNN and adopts an ensemble learning method to incorporate boosting weights. Thus, a fair comparison is to use ours as the base CNN in BoostCNN.

Table 5. Accuracies (%) on the CUB dataset. "bbox" and "parts" refer to object bounding box and part annotations.

methods	train anno.	test anno.	acc.
PB R-CNN [47]	bbox+parts	n/a	73.9
FG-Without [20]	bbox	bbox	82.0
SPDA-CNN [46]	bbox+parts	bbox+parts	85.1
STN [17]	n/a	n/a	84.1
B-CNN [24]	n/a	n/a	84.1
PDFS [48]	n/a	n/a	84.5
BoostCNN [30]	n/a	n/a	**85.6**
Ours	n/a	n/a	85.3

5.3.2 Results on the Aircraft and Cars datasets

The methods for the Aircraft and Cars datasets are all annotation-free since there are no ground-truth part annotations on these two datasets. We first evaluate our framework on the Aircraft dataset, and the related results are shown in the second column of Table 6. Our network achieves significantly better classification accuracy than the state-of-the-art B-CNN which can be seemed as a specific degree-2 case in our framework. As we find that *relu5_2* instead of *relu5_3* achieves the best performance in Aircraft dataset, our improvement might be due to the reasons: (1) B-CNN only focuses on *relu5_3* where the the discriminative parts are highly out-numbered, thus these parts might be overwhelmed by large non-discriminative region in pooling stage; (2) the discriminative parts in this dataset may occur simultaneously in both the coarse and fine scales. While the rich representation by incorporating multiple layers in our integration framework mitigates the local ambiguities of single-layer representation to a large extent.

The third column of Table 6 provides the comparision on the Cars dataset. B-CNN [24] shows the similar accuray behavior with ours and both present a large margin over Symbiotic [6] and FV-FGC [14]. The accuracy of B-CNN [24] using two networks is very close to ours (91.3% vs. 91.7%), yet for the single network case, it still has the accuracy gap of 1.1%, which infers that the hierarchical feature integration on a single network can also contribute the feature complementary as done by two different networks.

Table 6. Accuracies (%) on the Aircraft and Cars datasets.

methods	acc. (Aircraft)	acc. (Cars)
Symbiotic [6]	72.5	78.0
FV-FGC [14]	80.7	82.7
B-CNN [24]	84.1	91.3 (90.6)
Ours	**88.3**	**91.7**

5.3.3 Visualization for the learned image patches

In Fig. 4, we visualize some image patches with the highest activations in the deeper layers of our fine-tuned networks and the patches in each column come from different feature channels/maps. We obviously observe strong semantic-related parts such as heads, legs and tails in CUB; cockpit, tail stabilizers and engine in Aircraft; front bumpers, wheels and lights in Cars. Such observations exactly reflect the nature of our approach which aims to improve the feature discrimination by the effective combinations of these parts.

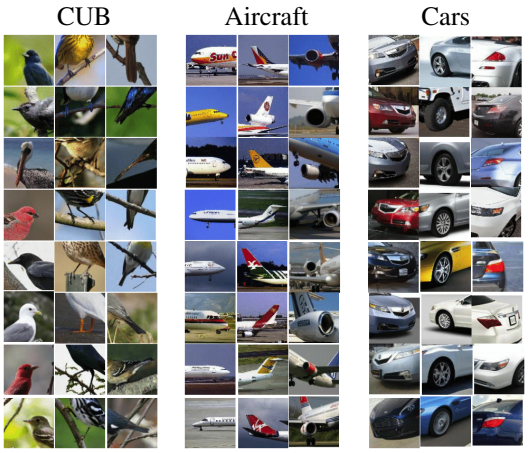

Figure 4. Visualization of the learned image patches in our fine-tuned networks on the CUB, Aircraft and Cars datasets.

6. Conclusion

It is preferred to perform FGVC under a more realistic setting without part annotations and any prior knowledge for explicit object appearance modeling. In this paper, by considering the weak parts in CNN itself, we present a novel higher-order integration framework of hierarchical convolutional layers to derive a rich representation for FGVC. Based on the kernel mapping scheme, we propose a polynomial predictor to exploit the higher-order part relations and presented the trainable polynomial modules which can be plugged in conventional CNNs. Furthermore, the higher-order integration framework can be naturally extended to mine the multi-scale part relations in hierarchical layers. The results on the CUB, Aircraft and Cars datasets manifest competitive performance, and demonstrate the effectiveness of our integration framework.

References

[1] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[2] A. Babenko and V. Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1269–1277, 2015.

[3] S. Branson, G. Van Horn, S. Belongie, and P. Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014.

[4] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision*, pages 438–451. Springer, 2010.

[5] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision*, pages 430–443. Springer, 2012.

[6] Y. Chai, V. Lempitsky, and A. Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 321–328, 2013.

[7] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3828–3836, 2015.

[8] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[9] M. Dixit, S. Chen, D. Gao, N. Rasiwasia, and N. Vasconcelos. Scene classification with semantic fisher vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2974–2983, 2015.

[10] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric p-norm feature pooling for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2609–2704. IEEE, 2011.

[11] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 317–326, 2016.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[13] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, pages 392–407. Springer, 2014.

[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.

[15] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[18] P. Kar and H. Karnick. Random feature maps for dot product kernels. In *AISTATS*, volume 22, pages 583–591, 2012.

[19] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[20] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5546–5555, 2015.

[21] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[22] Q. Le, T. Sarlós, and A. Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.

[23] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, volume 2, page 6, 2015.

[24] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[25] L. Liu, C. Shen, and A. van den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4749–4757, 2015.

[26] R. Livni, S. Shalev-Shwartz, and O. Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*, pages 855–863, 2014.

[27] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[28] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, pages 2627–2635, 2014.

[29] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[30] M. Moghimi, S. J. Belongie, M. J. Saberian, J. Yang, N. Vasconcelos, and L.-J. Li. Boosted convolutional neural networks. In *BMVC*, 2016.

[31] N. Murray and F. Perronnin. Generalized max pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2473–2480, 2014.

[32] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247. ACM, 2013.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[34] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1143–1151, 2015.

[35] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[38] D. Tao, X. Li, W. Hu, S. Maybank, and X. Wu. Supervised tensor learning. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[39] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[40] C. Wah, G. Van Horn, S. Branson, S. Maji, P. Perona, and S. Belongie. Similarity comparisons for interactive fine-grained categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–866, 2014.

[41] Y. Wang, J. Choi, V. Morariu, and L. S. Davis. Mining discriminative triplets of patches for fine-grained classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1163–1172, 2016.

[42] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015.

[43] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1798–1807, 2015.

[44] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2016.

[45] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

[46] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1143–1152, 2016.

[47] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[48] X. Zhang, H. Xiong, W. Zhou, W. Lin, and Q. Tian. Picking deep filter responses for fine-grained image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1134–1142, 2016.