# Task-Driven Feature Pooling for Image Classification

Guo-Sen Xie[†], Xu-Yao Zhang[†], Xiangbo Shu[§], Shuicheng Yan[‡], and Cheng-Lin Liu[†,⋆]
[†]NLPR, Institute of Automation, Chinese Academy of Sciences, China
[⋆]CAS Center for Excellence in Brain Science and Intelligence Technology, CAS, China
[‡]National University of Singapore, Singapore
[§]Nanjing University of Science and Technology, China
{guosen.xie, xyz, liucl}@nlpr.ia.ac.cn

## Abstract

*Feature pooling is an important strategy to achieve high performance in image classification. However, most pooling methods are unsupervised and heuristic. In this paper, we propose a novel task-driven pooling (TDP) model to directly learn the pooled representation from data in a discriminative manner. Different from the traditional methods (e.g., average and max pooling), TDP is an implicit pooling method which elegantly integrates the learning of representations into the given classification task. The optimization of TDP can equalize the similarities between the descriptors and the learned representation, and maximize the classification accuracy. TDP can be combined with the traditional BoW models (coding vectors) or the recent state-of-the-art CNN models (feature maps) to achieve a much better pooled representation. Furthermore, a self-training mechanism is used to generate the TDP representation for a new test image. A multi-task extension of TDP is also proposed to further improve the performance. Experiments on three databases (Flower-17, Indoor-67 and Caltech-101) well validate the effectiveness of our models.*

## 1. Introduction

Recent years have witnessed the rapid progress of visual recognition research, and more and more useful techniques have been developed, e.g., the traditional Bag of Words (BoW) model [5] and the recent benchmarking Convolutional Neural Network (CNN) model [18]. Both BoW and CNN models are considered as the milestones in the research community of object recognition. The BoW model consists of four steps, i.e., local feature (e.g., SIFT [23] and HOG [6]) extraction, feature coding [35], [32], coding vector pooling, and classifier training. The beginning of CNN is actually the same as the BoW model, but it differs from the latter in the following two aspects: (1) CNN is a fully supervised model in which all the parameters are trained in a discriminative manner; (2) CNN is a reduplicative convolution and pooling process until the last layer. The common part of BoW and CNN is that they both adopt the pooling operation in their architecture to increase the invariance of the representation.

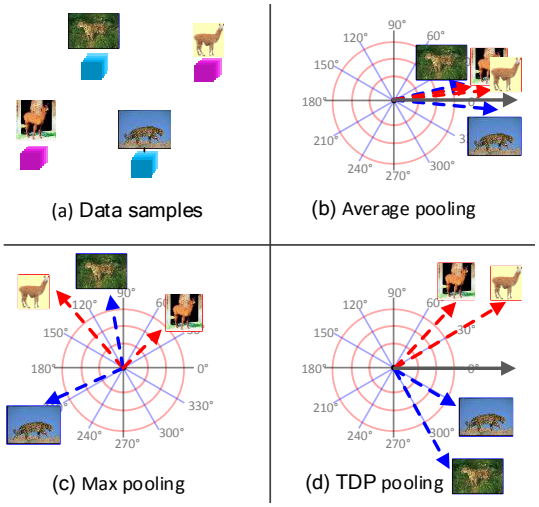


Figure 1. (a) Two confusing classes (Llama and Leopards). The tensors below the images are their corresponding representations, i.e., coding vectors from BoW or feature maps from CNN. The red and blue dashed arrows in (b) Average pooling, (c) Max pooling, and (d) the proposed TDP, are the pooled representations for Llama and Leopards respectively. It can be seen that TDP can well distinguish the two confused classes, while max and average poolings are difficult to separate the two classes. Best viewed in color.

Pooling is an important step in the BoW model, which can aggregate the input coding vectors into one vector and can also preserve spatial information when combined with the Spatial Pyramid Matching (SPM) [19] method. In the current CNN architecture, pooling also plays an important role. Max pooling is an essential step in original LeNet [20], current AlexNet [18], and Very Deep Nets (e.g., Vd16,

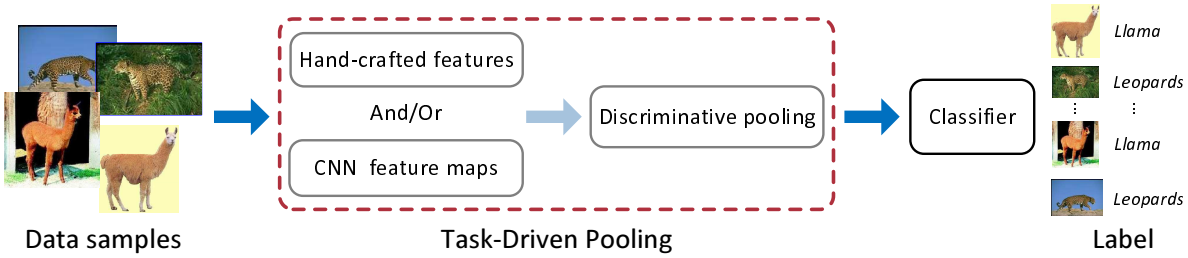


Figure 2. The pipline for image classification which incorporates our multiple task-driven pooling. Here procedure in red dashed box is our models, and the inputs for TDP (mTDP) learning are CNN feature maps of different scales. Best viewed in color.

Vd19, and GoogleNet) [30], [31]. Pooling operation in CNN can be seen as a kind of down-sampling, which can avoid the rotation variance to some extent.

The traditional pooling strategies include average pooling, max pooling [35], and generalized max pooling [24]. Average pooling simply takes the mean of the feature vectors over the whole (sub-region) of the images, making the pooled representations of the input images very similar to each other. Max pooling is obtained by taking maximum values from each dimension of feature vectors [35], while the max-pooled representation still has no supervision of image labels when it is being pooled. Recently, generalized max pooling [24] has been proposed to equalize the similarities between the feature vectors and the pooled representation yet it is still an unsupervised pooling method. Then, how to construct a discriminative pooling representation is an interesting topic.

Motivated by multi-task learning [4] and discriminative pooling learning [10], we take different feature extractors as different tasks. The purpose is to fuse the information under a unified framework to get a pooled representation. In this paper, we propose a new pooling strategy, termed as Task-Driven Pooling (TDP), which integrates representation learning and classifier training in a joint manner. The pooled representation is learned by equalizing the similarities between different descriptors and the pooled representation, and it can be seen as the bridge connecting the classifier and the pooling representation learning. Furthermore, a least square regression (LSR) classifier is trained jointly in the pooling process to transfer the supervised information into the representation learning process. The inputs of the TDP algorithm can be any features (e.g., different coding vectors of BoW and feature maps in CNN). After the optimization of TDP, we can get the pooled representations of all the training images. In the testing process, to generate TDP representation for a new image, we adopt a self-training strategy to efficiently solve the optimization problem. The differences between our proposed TDP model and traditional average and max pooling methods are shown in Figure 1. To further improve the performance, a multi-task extension of TDP (denoted as mTDP) is also proposed.

## 2. Related Work

In this section, we review the related works on image categorization and especially the pooling strategies utilized in recent pipelines. From the BoW model [5] to the CNN model [18], there have been tremendous developments in the field of image categorization, which also boost the advances in other relevant areas, such as object detection [14].

**Bag of Words Model.** The BoW model is a standard and the best pipeline before the successful application of CNN on large scale image categorization problems [18]. Let $I$ be an input image. The BoW model consists of four steps: (1) local feature extraction on $I$, to obtain the local descriptor tensor $A \in \mathbb{R}^{d\times l\times w}$, where $d$ is the descriptor dimension, e.g., $d = 128$ for SIFT [23] descriptor, and $l$, $w$ are the length and the width of the tensor; (2) local feature coding, resulting in the coding vector tensor $V \in \mathbb{R}^{K\times l\times w}$, where $K$ is the coding vector dimension; (3) pooling of the coding vector tensor $V$, leading to a pooled representation $v \in \mathbb{R}^{K\times 1}$, with the dimension the same as the original coding vector; and finally (4) classifier training (e.g., linear SVM classifier) on $v$. Before the pooling step, we can also incorporate SPM [19] to capture the spatial information for better performance. The popular coding methods include hard coding [5], soft coding [22], sparse coding [35], locality-constrained linear coding [32], etc. Other methods include Fisher vector coding [26] and super vector coding [40], which are based on calculating the first and the second order differences between descriptors and their clustering centers.

**Convolutional Neural Network Model.** Since the CNN model gained the champion in the large-scale competition of ImageNet classification [18] in 2012 (ILSVRC-12), it is becoming the dominant model in visual recognition. The main components of CNN are multiple convolution, pooling and different activating functions. Actually, the first convolutional and pooling layer can be seen as the BoW model (by viewing filters of CNN as the dictionary elements in BoW, and the convolution between filters and input image as feature coding in BoW). This motivated many researchers to construct deep BoW models to explore the gap between BoW and CNN [1]. As we know, CNN is a fully

supervised model while BoW is a step-by-step model and most steps are unsupervised. This is the main difference between BoW and CNN. Recently, the very deep models (Vd16, Vd19 and GoogleNet [30], [31]) have achieved the best performance in the image categorization task. Considering the large number of different CNN models, we can integrate different feature informations contained in CNNs to achieve better performance. The most useful feature extractors of the CNN are the feature maps in the middle layers and the fully connected representations in the last several layers [37]. Here, we denote the CNN feature maps as $V \in \mathbb{R}^{K\times h\times h}$, where $K$ is the feature map number, and $h$ is the size of all these maps. These maps cannot be directly used for classification, and we need some operation on them to get the final representation, which is beneficial for classification. Such operation can be the pooling similar as that on the coding vectors in the BoW model. Here in this paper, we aim to process different scaled feature maps of one CNN in a unified framework.

**Recent Pooling Methods.** Pooling is to aggregate tensor $V \in \mathbb{R}^{K\times l\times w}$ (e.g., feature maps from CNN) into one representation $v \in \mathbb{R}^K$, which can be fed into the classifier for training. Pooling has been used in many visual recognition systems, e.g., average pooling used in the biologically-inspired visual systems [13], and max pooling [29] used in the CNN [18]. An obvious drawback of average pooling is that the total $l \times w$ coding vectors from the tensor $V \in \mathbb{R}^{K\times l\times w}$ are simply averaged on each dimension without any consideration of the importance of different coding vectors [3]. While for max pooling, it is the maximum response from each dimension of $V$, without any supervision on whether the max-pooled representation is beneficial for afterward classification or not. Weighted max pooling [10] has been proposed by giving larger weights to more important coding vectors corresponding to foreground image regions. On the other hand, close coding vectors from $V$ should be pooled together [2], wherein the notion of closeness is in both the geometric and descriptor domains. Feng et al. [12] proposed geometric $l_p$-norm pooling (GLP), which is aimed at learning $K$ ($V \in \mathbb{R}^{K\times l\times w}$) weighted vectors for all the codewords by taking the class separability as the objective function, thus GLP is very time consuming. Recently, generalized max pooling (GMP) [24] was proposed for coding vector pooling, wherein the pooled representation of the whole image or the sub-region of the image is optimized by equalizing the similarity between each coding vector and the pooled representation. GMP is generic and can be applied to all aggregation-based coding vector pooling problems through optimization. However, GMP is still an unsupervised method for feature pooling. Our TDP (mTDP) in this paper is formulated as the joint (multiple) classifier training and pooling representation generation. Figure 2 illustrates the whole pipline for image classification.

## 3. Task-Driven Pooling

To integrate different feature descriptors in a joint and discriminative manner, we propose the task-driven pooling (TDP), by combining the representation learning and classifier training in a unified framework. Let $V \in \mathbb{R}^{K\times N}$ be the input data, where $N = l \times w$, e.g., $K = 512, N = 8 \times 8$ for some CNN feature maps.

### 3.1. Single Task-driven Pooling

Given $m$ training images, the corresponding input is $\{(V_i, y_i)\}_{i=1}^m$, where $V_i \in \mathbb{R}^{K\times N}$, $i = 1, 2, \cdots, m$ and $y_i \in \{1, 2, \cdots, C\}$ with $C$ being the number of classes. We use least square regression (LSR) for the multiclass classification purpose. Let $F = [f_{y_1}, f_{y_1}, \cdots, f_{y_m}]^T \in \mathbb{R}^{m\times C}$ be the label matrix of all the training images, where $f_j = [0, \cdots, 0, 1, 0, \cdots, 0]^T \in \mathbb{R}^C$ is an all-zero vector except the $j$-th element which is one.

Our goal is to learn the pooled representation from the input $V \in \mathbb{R}^{K\times N}$ (which can be the coding vectors of BoW or feature maps of CNN). We use $\{\varphi_i \in \mathbb{R}^{K\times 1}\}_{i=1}^m$ to represent the learned representations for all the training images. Let $\Phi = [\varphi_1, \varphi_2, \cdots, \varphi_m]^T \in \mathbb{R}^{m\times K}$ be the matrix format of all the TDP representations. Furthermore, let the classifier parameters of LSR be $W \in \mathbb{R}^{K\times C}$ and $b \in \mathbb{R}^{C\times 1}$. Therefore, the task-driven pooling (TDP) problem can be formulated as the following optimization problem:

$$\begin{aligned}\min_{W,\Phi,b} &\|\Phi W + \mathbf{1}_m b^T - F\|_F^2 + \mu\|W\|_F^2 \\ &+ \lambda\Sigma_{i=1}^m(\|V_i^T\varphi_i - \mathbf{1}_N\|_2^2 + \gamma\|\varphi_i\|_2^2),\end{aligned} \tag{1}$$

where $\mathbf{1}_m \in \mathbb{R}^{m\times 1}$ and $\mathbf{1}_N \in \mathbb{R}^{N\times 1}$ are the vectors with all the elements being ones, and $\lambda$, $\gamma$ and $\mu$ are the hyper-parameters of our model.

The first term of this model is a linear classification model with the learned pooled representation $\varphi_i$ as the input, where the regularization terms (last two terms of the model) are to guarantee the pooled representation $\varphi_i$ to be close to each column of the original coding vector $V_i$. Similar as [24], by using this regularization, the learned representation can equalize the similarities between $\varphi_i$ and different columns of $V_i$ by forcing their inner product to be exactly one. In this way, we can utilize more information in $V_i$ to get a task-driven pooled representation $\varphi_i$ which is more suitable for the classification task. Note that TDP is an implicit pooling method where we do not define the explicit pooling function. Furthermore, each image has its own pooling process according to the specific classification task. This strategy should be much more flexible and accurate than the pre-defined pooling strategy such as max and average poolings.

The formulation in (1) is convex w.r.t. $W, \Phi, b$ separately, thus the alternating optimization can be used to solve $(W, b)$ and $\Phi$, i.e., optimizing one while the other is fixed. Note that closed-form solutions can be obtained for both the two subproblems, which makes the optimization process much easy and stable. The detailed procedures are as following:

**Pooling Representation Learning.** Here we fix $(W, b)$, and solve $\Phi$. The model (1) can be changed into:

$$\begin{aligned}\min_{\Phi} g(\Phi) =&\|\Phi W + \mathbf{1}_m b^T - F\|_F^2 \\ &+ \lambda \Sigma_{i=1}^m (\|V_i^T \varphi_i - \mathbf{1}_N\|_2^2 + \gamma \|\varphi_i\|_2^2).\end{aligned} \tag{2}$$

$g(\Phi)$ in (1) can be further equalized as:

$$\begin{aligned}g(\Phi) =&\Sigma_{i=1}^m \{\|W^T \varphi_i + b - F_{i,\cdot}^T\|_F^2 \\ &+ \lambda(\|V_i^T \varphi_i - \mathbf{1}_N\|_2^2 + \gamma \|\varphi_i\|_2^2)\},\end{aligned} \tag{3}$$

where $F_{i,\cdot} \in \mathbb{R}^{1\times C}$ is the $i$-th row of the label matrix $F$.

Let the derivatives of $g(\Phi)$ w.r.t. $\varphi_i$ be zero, and we can get the following closed-form solution of $\varphi_i, i = 1, 2, \cdots, m$.

$$\varphi_i = [WW^T + \lambda V_i V_i^T + \lambda\gamma \mathbf{I}_K]^{-1}[W(F_{i,\cdot}^T - b) + \lambda V_i \mathbf{1}_N], \tag{4}$$

where $\mathbf{I}_K \in \mathbb{R}^{K\times K}$ is the identity matrix.

**LSR Learning.** By fixing the learned $\Phi$ in the last step, we solve $(W, b)$ in this part. The optimization in (1) is changed into the classical LSR problem:

$$\min_{W,b} q(W, b) = \|\Phi W + \mathbf{1} b^T - F\|_F^2 + \mu \|W\|_F^2. \tag{5}$$

The LSR problem in (5) has closed-form solution [33] w.r.t. $(W, b)$. The optimal solution of (5) can be calculated as

$$W = (\Phi^T H \Phi + \mu \mathbf{I}_K)^{-1} \Phi^T H F, \tag{6}$$

$$b = \frac{(F - \Phi W)^T \mathbf{1}_m}{m}, \tag{7}$$

where $H = \mathbf{I}_m - \mathbf{1}_m \mathbf{1}_m^T / m$.

After repeating the above two sub optimization problems for multiple times, we can get the solution of the original TDP problem. In this way, both the pooled representations for all the training images and the classifier parameters can be obtained simultaneously.

### 3.2. Self-training for Testing Image

After optimization of (1), the pooled TDP representations $\Phi = [\varphi_1, \varphi_2, \cdots, \varphi_m]$ for all the $m$ training samples are learned at the same time. However, in the testing process, all the samples are unlabeled, we cannot directly learn their pooled representations from (1), and thus we use a self-training mechanism to learn the predictions and representations simultaneously.

Given $n-m$ testing samples denoted as $\{(V_i, y_i)\}_{i=m+1}^n$, where $V_i \in \mathbb{R}^{K\times N}$, $i = m+1, m+2, \cdots, n$ and $\{y_i\}_{i=m+1}^n$ are the labels which need to be predicted, we first calculate the initial representations $\{\varphi_i \in \mathbb{R}^{K\times 1}\}_{i=m+1}^n$ of $V_i$, $i = m+1, m+2, \cdots, n$ based on the following formulation [24]

$$\min_{\varphi} \|V_i^T \varphi - \mathbf{1}_N\|^2 + \gamma \|\varphi\|^2. \tag{8}$$

Solving (8) for $i = m+1, m+2, \cdots, n$ leads to

$$\varphi_i = (V_i V_i^T + \gamma \mathbf{I}_K)^{-1} V_i \mathbf{1}_N, \tag{9}$$

where $\gamma$ is the same as the corresponding parameter of (1) during the training phase. Then the predicted label of $V_i$ can be calculated from both $\varphi_i$ (Eqn. (9)) and the learned LSR parameter $(W, b)$ as follows

$$\hat{F}_{i,\cdot} = \varphi_i^T W + b^T. \tag{10}$$

Since $\hat{F}_{i,\cdot}$ is a soft prediction, we lead to quantize it into an 0-1 vector for the purpose of LSR learning in (1). Let $l_i = \arg\max_{j=1}^C \hat{F}_{i,j}$, $i = m+1, \cdots, n$. We can further define the 0-1 vector w.r.t. $V_i$ as follows

$$\breve{F}_{i,\cdot} = [\underbrace{0\ 0\ \cdots\ 0}_{l_i - 1}\ 1\ \underbrace{0\ \cdots\ 0}_{C - l_i}] \in \mathbb{R}^{1\times C}, \tag{11}$$

where the element "1" is located on the $l_i$-th position.

To generate discriminative representations of testing images, we take $\hat{F}_{i,\cdot}, i = m+1, \cdots, n$ as the initializing labels. With fixed LSR parameter $(W, b)$ and model parameters $\gamma, \mu$[1], we alternately update $\hat{F}_{i,\cdot}$ and $\varphi_i$ $(i = m+1, \cdots, n)$ based on Eqn. (4) and (10). Note that for the purpose of LSR training, in updating of Eqn. (4), $F_{i,\cdot}$ is replaced by $\breve{F}_{i,\cdot}$ in Eqn (11). In this way, the hyper-parameters for generating final pooling representations of testing images are the number of iterations (fixed as 10) and $\lambda$ when updating Eqn. (4) and (10).

After updating Eqn. (4), (10) and (11) repeatedly, we take the obtained $\{\varphi_i\}_{i=m+1}^n$ as our final pooled representations for testing images. With the learned pooled representations, we can feed them into the classifier for afterward classification. This is a self-training process, where we first deduce an unreliable prediction, then use it to learn the pooled representation, and after that the previous prediction is refined by the classifier with the new pooled representation as the input. This process is repeated to boost the accuracy gradually. Self-training has been widely used in semi-supervised learning, and here we use it to generate the TDP representation for a new testing image efficiently and effectively.

[1] $\gamma, \mu$ are the same as the ones used in (1).

### 3.3. Multiple Task Extension of TDP

To further improve the performance, we also propose a multiple task extension of the TDP model (mTDP). We first define the definitions and notations related to mTDP, and then describe the detailed algorithm of mTDP (Algorithm 1).

Suppose $T$ feature descriptors (tasks) exist for both the $m$ and $n-m$ training and testing images, which are denoted as $\{V_i^{(t)}, y_i\}_{i=1}^n, t = 1, 2, \cdots, T$, where $V_i^{(t)} \in \mathbb{R}^{K_t \times N_t}$, $y_i \in \{1, 2, \cdots, C\}$, with $C$ being the number of classes. $\{y_i\}_{i=m+1}^n$ denotes the labels of testing images which need to be predicted. The $K_t, N_t$ are the dimension and the vector number of the input samples for the $t$-th tasks. We only consider the case of $K_1 = K_2 = \cdots = K_T = K$ in this paper[2]. For $T$ tasks of $m$ training images, $F \in \mathbb{R}^{m \times C}$ is their label matrix, the same as the single TDP. Similarly, denote $\{\varphi_i^{(t)} \in \mathbb{R}^{K \times 1}\}_{i=1}^n$ as the TDP representations of $n$ images for the task $t$. Let $\Phi^{(t)} \in \mathbb{R}^{m \times K}$ and $\hat{\Phi}^{(t)} \in \mathbb{R}^{(n-m) \times K}$ be the matrix format of TDP representation of the task $t$ for training and testing images respectively. Finally, let the parameters of LSR for the task $t$ be $W^{(t)} \in \mathbb{R}^{K \times C}$ and $b^{(t)} \in \mathbb{R}^{C \times 1}$. Now the mTDP can be formulated as

$$\min_{\{W^{(t)}, \Phi^{(t)}, b^{(t)}\}_{t=1}^T} \sum_{t=1}^{T} \Psi(W^{(t)}, \Phi^{(t)}, b^{(t)}) + \beta \Omega(\boldsymbol{W}), \quad (12)$$

where $\Psi(W^{(t)}, \Phi^{(t)}, b^{(t)})$ is the single TDP formulation for the task $t$, and is defined as

$$\begin{aligned} \Psi = &\|\Phi^{(t)} W^{(t)} + \mathbf{1}_m b^{(t)^T} - F\|_F^2 + \mu \|W^{(t)}\|_F^2 \\ &+ \lambda \Sigma_{i=1}^m (\|V_i^{(t)T} \varphi_i^{(t)} - \mathbf{1}_{N_t}\|_2^2 + \gamma \|\varphi_i^{(t)}\|_2^2). \end{aligned} \quad (13)$$

Moreover, $\boldsymbol{W} = [W^{(1)}, W^{(2)}, \cdots, W^{(T)}] \in \mathbb{R}^{K \times TC}$ is the multi-task concatenated matrix, and $\Omega(\boldsymbol{W})$ can reflect the shared information while learning all the task parameters, which is defined as follows [4]

$$\Omega(\boldsymbol{W}) = tr((\mathbf{W}^T \mathbf{W})^{\frac{1}{2}}). \quad (14)$$

The whole optimizing procedure of mTDP is listed in Algorithm 1. After learning $\{W^{(t)}, \Phi^{(t)}, b^{(t)}\}_{t=1}^T$ for training samples, we conduct pooling representation learning of testing samples for each task, which is the same as in Section 3.2.

## 4. Experiments

We conduct experiments on three databases (fine-grained, scene and object recognition databases), i.e., Flower-17 [25], Indoor-67 [27] and Caltech-101 [11]. We first describe the databases and experimental settings, and then report the results.

[2]When $K_1 \neq K_2 \neq \cdots \neq K_T$, we can turn them into equalities by dimension reduction methods.

**Algorithm 1** Multiple Task-Driven Pooling

**Input:** Training data: $\{V_i^{(t)}\}_{i=1}^m, t = 1, 2, \cdots, T$, labels: $F$, and $\lambda, \gamma, \mu$ for $T$ tasks, $\beta$, *IterNum*.
**Output:** LSR parameters $\{W^{(t)}, b^{(t)}\}_{t=1}^T$ and the pooling representations $\{\Phi^{(t)}\}_{t=1}^T$.
**Initialization:** $\{\Phi^{(t)}, W^{(t)}, b^{(t)}\}_{t=1}^T$, based on TDP (See (1)), and reserve corresponding model parameters $(\lambda, \gamma, \mu)$,$s = 1$. $H = \mathbf{I}_m - \frac{1}{m}\mathbf{1}_m \mathbf{1}_m^T$.

**while** $s \leq$ *IterNum* **do**
  $\mathbf{W} = [W^{(1)}, W^{(2)}, \cdots, W^{(T)}]$.
  **for** $t = 1 \rightarrow T$ **do**
    **Optimize** $\Phi^{(t)}$ of Eqn. (13) with fixed $(W^{(t)}, b^{(t)})$.
    **for** $i = 1 \rightarrow m$ **do**
      $\varphi_i^{(t)} = [W^{(t)} W^{(t)^T} + \lambda V_i^{(t)} V_i^{(t)^T} + \lambda\gamma \mathbf{I}_K]^{-1} [W^{(t)} (F_{i,\cdot}^T - b^{(t)}) + \lambda V_i^{(t)} \mathbf{1}_{N_t}]$.
    **end for**
    $\Phi^{(t)} = [\varphi_1^{(t)}, \varphi_2^{(t)}, \cdots, \varphi_m^{(t)}]^T$.
    **Optimize** $(W^{(t)}, b^{(t)})$ of Eqn. (12) with fixed $\Phi^{(t)}$.
    $W^{(t)} = (\Phi^{(t)^T} H \Phi^{(t)} + \mu \mathbf{I}_K + \frac{\beta}{2} (\mathbf{W}\mathbf{W}^T)^{-\frac{1}{2}})^{-1} \Phi^{(t)^T} H F$.
    $b^{(t)} = \frac{(F - \Phi^{(t)} W^{(t)})^T \mathbf{1}_m}{m}$.
  **end for**
  $s \leftarrow s + 1$
**end while**

### 4.1. Datasets and Settings

We use the standard training-test partitions. Specifically, for Flower-17, we adopt the existing three splits of training-test sets, 40 images per class for training and 20 images per class for testing. For the Indoor-67 database, we use the partition of training-test the same as [27]: about 80 images per class for training and 20 images per class for testing. For Caltech-101 we randomly generate three partitions for training and test sets, so that each partition contains 30 training images per class, and up to 50 test images per class [16], [1]. Performance on the three datasets is evaluated by the average class accuracy.

**Flower-17 Database.** The Flower-17 database [25] is a fine-grained dataset, which contains 17 categories with 80 images per class. The scale, pose, and lighting variations are very large in this dataset. The inter-class difference is very little, but the intra-class similarity is very large.

**MIT Indoor-67 Database.** The Indoor-67 database [27] is a popular indoor scene database, including 15,620 images of 67 indoor scenes. It is very difficult to distinguish different classes, because the categories are all indoor scenes, and inter-class variance between different classes is very little.

**Caltech-101 Database.** The Caltech-101 database [11] is an image classification database, which contains totally 9,144 images of 102 categories (101 object categories and one background category), with 31 to 800 images per class. It is a database which has been used for testing many algorithms in recent years, including both the BoW models and CNN models.

**The Definitions of Multiple Tasks.** We take feature maps of different scales from the same CNN as different tasks, which are used as inputs of our models. The feature maps from the last convolutional layer of the Very Deep 19 Net (Vd19) [30] are used. Specifically, for Flower-17 and Indoor-67, feature maps of two scales (seen as two tasks) are generated by forward propagating the input images of size $224 \times 224$ and $384 \times 384$, thus the generated two kinds of input tasks for the $i$-th image are $\{V_i^{(1)}, V_i^{(2)}\}$ where $V_i^{(1)} \in \mathbb{R}^{512\times196}$ and $V_i^{(2)} \in \mathbb{R}^{512\times576}$. Similarly, for Caltech-101, feature maps of three scales (seen as three tasks) for the $i$-th image are $\{V_i^{(1)}, V_i^{(2)}, V_i^{(3)}\}$ and $V_i^{(1)} \in \mathbb{R}^{512\times256}$, $V_i^{(2)} \in \mathbb{R}^{512\times576}$, and $V_i^{(3)} \in \mathbb{R}^{512\times1024}$ respectively. Here the scales of input images in extracting the three input tasks for Caltech-101 are $256 \times 256, 384 \times 384, 512 \times 512$. Moreover, we use the caffe implementation [17] of Vd19 to extract the CNN feature maps, and the fully connected layer activates, which can get enhanced performance on both databases when combined with our TDP or mTDP representations.

Note that besides the use of the 4096-dimensional vectors (abbreviated as "fc") from the penultimate fully connected layer of Vd19 Net, to combine different network information, we also extract the penultimate fully connected layer representations (abbreviated as "fc-2") of Vd16 Net [30], both of them are with relu [18]. In this paper, "fc", "fc-2" and the pooled representations of all the methods are first $L2$-normalized and then fed into the linear SVM classifier [9] for classification.

**Parameter Settings.** During the training phase of our TDP model, the parameter $\lambda$ is set as 1 in all the cases. $\gamma$ is empirically selected from $\{5, 10, 15, 20\}$, and $\mu$ is from $\{0.05, 0.1, 0.2, 1\}$. The number of iterations is fixed as 10. As for the training of the mTDP model, we use the parameters which are the same as their corresponding single TDP models, and $\beta \in \{0, 1, 2, 3, 4\}$. We will analyze the influence of $\beta$ on the classification rates of the mTDP model in Section 4.5. As for the parameter $C$ for the linear SVM, we fix it as 1 for Flower-17 and Indoor-67, and 1,000 for Caltech-101 respectively.

## 4.2. Flower-17 Experiments

In this part, we first show the two single task comparisons of our TDP model, denoted as sTDP, with other state-of-the-art pooling methods (i.e., average, max and gmp poolings) under the same protocols. Besides comparisons of different pooling methods directly, we also compare the performances when combined them with the penultimate fully connected layer activates "fc". Here the "fc" representations under two single task settings are the same, by forward propagating resized original images until the penultimate layer of Vd19 net. It can be concluded from Table 1 that our sTDP representations are not only much better than other pooling methods, but also can boost the performance of "fc" better than their counterparts. Note that the result of sTDP under task-2 in Table 1 is $92.65\%$ with dimension 512, which have already surpassed the performance of "fc" (with dimension 4096).

| Method | Single task1 | Single task2 | Multiple task1+task2 |
|---|---|---|---|
| Ave | 87.55±0.45 | 86.08±0.45 | 89.61±0.90 |
| Max | 87.65±0.29 | 89.61±0.90 | 91.08±0.85 |
| Gmp | 88.24±1.06 | 88.92±0.34 | 90.20±0.45 |
| sTDP/mTDP | **92.35±1.28** | **92.65±0.59** | **92.74±0.34** |
| fc | 92.35±0.88 | 92.35±0.88 | 92.35±0.88 |
| Ave+fc | 92.75±1.62 | 92.75±1.33 | 93.14±1.91 |
| Max+fc | 92.65±1.35 | 93.53±1.35 | 93.24±0.78 |
| Gmp+fc | 92.55±1.33 | 93.04±1.03 | 93.33±1.22 |
| sTDP/mTDP+fc | **93.73±1.03** | **93.92±1.03** | **94.12±1.28** |

Table 1. The classification rates of different pooling representations, and their combined representations with "fc", under two single task and the multiple task settings on Flower-17.

| Method | Acc (%) |
|---|---|
| Visual vocabulary [21] | 71.76±1.76 |
| BRD [34] | 89.06±0.60 |
| GRLF [36] | 91.70±1.70 |
| Vd19 (fc) | 92.84±1.11 |
| sTDP-task2+fc | **93.92±1.03** |
| mTDP+fc | **94.12±1.28** |
| mTDP+fc+fc-2 | **94.80±0.90** |

Table 2. Comparisons of the classification rates of our methods with other state-of-the-art methods on Flower-17.

To further evaluate the performance under the multiple task setting of our model (mTDP), we also compare the cascaded representations of Max, Ave, and Gmp with the proposed mTDP representation, which are optimized by incorporating the shared information between different inputs (tasks). See Table 1 for the detailed results, which again reflect the fact that our mTDP outperforms all the other methods.

Finally, comparisons with other methods are listed in Table 2. It can be seen that competitive results have been achieved with low dimensional pooled features, e.g., the result of $93.92\%$ is obtained based on the linear SVM classifier with the sample dimension being $4096 + 512$.

## 4.3. Indoor-67 Experiments

We report the experimental results on Indoor-17 in this part. We first show comparisons of two single task cases of our TDP (sTDP) with other methods (average, max and gmp poolings). The performance comparisons of representations (sTDP, ave, max and gmp) combined with "fc" are also considered. It can be seen from Table 3 that our sTDP representation outperforms other methods in all the settings. Moreover, we list the accuracies of the combination of different tasks in Table 3. It can be seen that our methods can get better results than the counterparts, and can boost the performance of "fc" greatly.

To compare with other state-of-the-art methods on this database, we list the results in Table 4. Our performance

| Method | Single task1 | Single task2 | Multiple task1+task2 |
|---|---|---|---|
| Ave | 63.67 | 63.21 | 67.62 |
| Max | 61.38 | 63.38 | 65.84 |
| Gmp | 64.48 | 64.09 | 66.62 |
| sTDP/mTDP | **67.38** | **68.07** | **69.65** |
| fc | 72.96 | 72.96 | 72.96 |
| Ave+fc | 73.54 | 73.57 | 74.10 |
| Max+fc | 73.11 | 74.13 | 73.64 |
| Gmp+fc | 73.32 | 73.08 | 73.57 |
| sTDP/mTDP+fc | **74.38** | **74.86** | **75.61** |

Table 3. The classification rates of different pooling representations, and their combined representations with "fc", under two single task and the multiple task settings on Indoor-67.

| Method | Acc (%) |
|---|---|
| Mode-seeking+IFV [7] | 66.87 |
| ISPR+IFV [21] | 68.50 |
| Order-less pooling [15] | 68.90 |
| CNN-aug+SVM [28] | 69.00 |
| Hybrid-net+fc [39] | 70.08 |
| Vd19 (fc) | 72.96 |
| mTDP+fc | **75.61** |

Table 4. Comparisons of the classification rates of our methods with other state-of-the-art methods on Indoor-67.

| Method | Single task1 | Single task2 | Single task3 |
|---|---|---|---|
| Ave | 83.69±0.61 | 81.16±0.95 | 77.69±0.41 |
| Max | 86.04±0.65 | 84.61±1.04 | 82.39±0.59 |
| Gmp | 85.83±0.43 | 85.15±0.45 | 80.61±0.46 |
| sTDP | **88.72±0.82** | **87.92±0.59** | **85.51±0.56** |
| fc | 91.90±0.37 | 91.90±0.37 | 91.90±0.37 |
| Ave+fc | 92.05±0.71 | 92.15 ±0.74 | 92.06±0.88 |
| Max+fc | 92.53±0.79 | 92.62±0.15 | 92.64±0.53 |
| Gmp+fc | 92.13±0.89 | 92.27±0.73 | 92.26±0.64 |
| sTDP+fc | **92.68±0.18** | **92.85±0.33** | **92.72±0.27** |

Table 5. The classification rates of different pooling representations, and their combined representations with "fc", under three single task settings on Caltech-101.

is much better than the traditional methods, such as mode seeking [7], and is competitive with some CNN based methods, e.g., the order-less pooling [15].

### 4.4. Caltech-101 Experiments

In this part, we conduct three single task comparisons of our TDP model with other pooling methods under the same protocols. From Table 5, it can be seen that our sTDP is better than other methods consistently.

Three tasks (t1,t2 and t3) can have four combinations, i.e., t1+t2, t2+t3, t3+t1 and t1+t2+t3, for each pooling method. We show the four combinations and their combinations with "fc" in Figure 3 (from "1" to "8" it is the representation of t1+t2, t2+t3, t3+t1, t1+t2+t3, $\cdots$, "fc"+t1+t2+t3 respectively ). Our mTDP representations can perform better than other counterpart ones, almost in all the cases. It seems that in the case of object recognition the combinations of our mTDP representations with the "fc" layer are just a little better than their corresponding max pooling relevant combinations (see "5"-"8" in Figure 3). Possible reasons for this phenomenon may be that Caltech-101 is an object-oriented database, thus max pooled representation from the convolutional layer of CNN can obtain

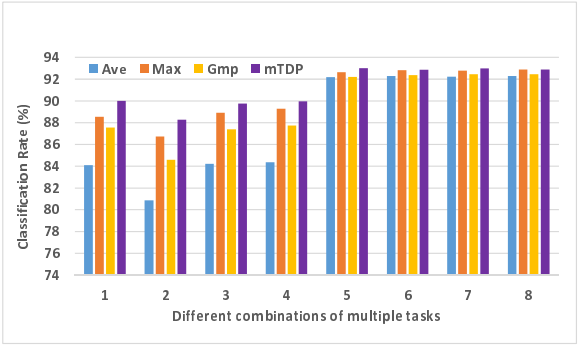


Figure 3. The classification rates of different combinations of multiple tasks and their combinations with "fc" on Caltech-101 database.

| Method | Acc (%) |
|---|---|
| ScSPM [35] | 73.20±0.54 |
| LLC-SPM [32] | 73.44 |
| Multipath Deep BoW [1] | 82.50±0.50 |
| Visualize CNN [38] | 86.50±0.50 |
| DeCAF-fc6 [8] | 86.91±0.70 |
| VGG-Vd16 [30] | 91.80±1.00 |
| VGG-Vd19 [30] | 92.30±0.50 |
| VGG-Vd16+Vd19 [30] | 92.70±0.50 |
| SPP-CNN [16] | 93.42±0.50 |
| mTDP+fc | 93.02±0.24 |
| mTDP+fc+fc-2 | **93.68±0.50** |

Table 6. Comparisons of the classification rates of our methods with other state-of-the-art methods on Caltech-101.

more large activates w.r.t. the object in the image. Then the representations are also complementary with the "fc" representation. On contrary, for scene or flower images where the layout is almost global, we have found that mTDP can enhance "fc" most (see the experiments in Section 4.2-4.3). So we can conclude that flower and scene recognition are more suitable for our TDP representation than object recognition problems.

Finally, comparisons with other state-of-the-art models are listed in Table 6. Note that mTDP representation combined with "fc" and "fc-2" can get the best result of $93.68\%$ on this database so far, with lower dimension of $4096 \times 2 + 512 \times 3$.

### 4.5. Parameter and Convergence Analysis

We take Indoor-67 as an example to analyze the parameters and convergence of sTDP and mTDP algorithms. In all experiments, we only iterate 10 times of the algorithms. The whole training/test times of sTDP for the three datasets are less than 10 minutes (554/147s for Indoor-67, 442/457s for Caltech-101, and 75/45s for Flower-17). We draw the convergence curve both on two single task and the multiple task settings. Our TDP algorithms are alternative optimization methods, in each iteration of updating parameters, there exist closed-form solutions of the parameters. From the convergent curve in Figure 4, it can be concluded that our algorithms can reach convergence within 10 iterations, which reduces the time consumption dramatically. To un-

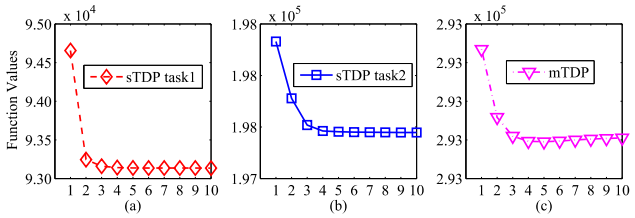


Figure 4. The convergence curves w.r.t. the iterating of our algorithm during training phase on Indoor-67.

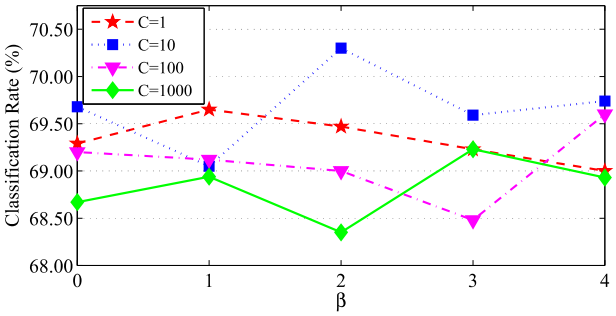


Figure 5. "C" denotes the linear SVM parameters, x-axis means the values of $\beta$ and y-axis is the classification rate.

derstand the influence of our mTDP model (Algorithm 1) w.r.t. parameter $\beta$ when other parameters are fixed, we draw the curve of mTDP accuracy w.r.t. $\beta$ under different values of $C$ for training the linear SVM (Figure 5). From Figure 5, a conclusion can be drawn that the classification rate is changing w.r.t. $\beta$, and the best result is obtained when $\beta = 1$ when the linear SVM parameter $C$ is taken as 1.

### 4.6. Visualization

Given the input $V \in \mathbb{R}^{K\times N}$ for one image, and the learned TDP (Gmp) representation of $V$ is $\varphi \in \mathbb{R}^{K\times 1}$. Then $\varphi$ can be written as the linear combination of all the columns of $V$ [24], i.e., $\varphi = V\alpha$, where $\alpha$ can be seen as the weights of weighted pooling. We calculate $\alpha$ as follows:

$$\alpha = (V^T V + I)^{-1} V^T \varphi. \tag{15}$$

Visualization of $\alpha$ for some given images is illustrated in Figure 6. It can be seen that (1) the weights of TDP can address more discriminative regions (red regions) in scene images, and (2) the weights of TDP can cover more object parts in object images.

### 4.7. TDP under Fine-tuned CNN

In this part, we discuss the effectiveness of TDP under fine-tuned CNN. Taking Indoor-67 as an example, we fine-tune the VGG-Vd19 [30] under Caffe [17]. The initialization of the parameters is the same as the model trained on ImageNet database except that the parameters of the last layer are randomly initialized with Gaussian distribution. The learning rates of all the lower layers are initialized as 0.0001, except that the last fully-connected layer is initialized as 0.001. We iterate the fine-tuning about 120 epochs

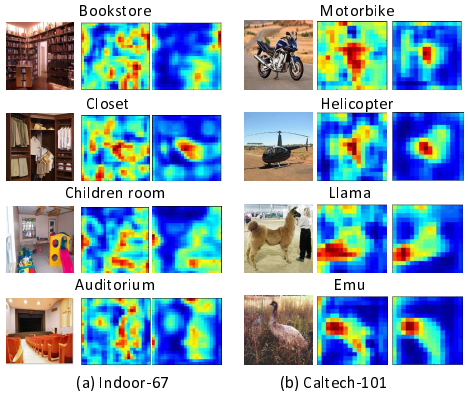


Figure 6. Visualization of weights $\alpha$ (Eqn. 15) on (a) Indoor-67 and (b) Caltech-101 database. Here the 2-th and 3-th column in (a) and (b) are the visualizations of TDP, and Gmp respectively.

| Method | Single task1 | Single task2 | Multiple task1+task2 |
|---|---|---|---|
| Ave | 68.14 | 65.28 | 70.74 |
| Max | 67.44 | 67.33 | 71.56 |
| Gmp | 68.08 | 67.18 | 71.32 |
| sTDP/mTDP | **70.84** | **70.88** | **72.99** |
| fc | 74.39 | 74.39 | 74.39 |
| Ave+fc | 74.77 | 74.83 | 74.98 |
| Max+fc | 74.95 | 75.34 | 75.38 |
| Gmp+fc | 74.85 | 74.91 | 74.69 |
| sTDP/mTDP+fc | **75.65** | **76.19** | **76.31** |

Table 7. The classification rates of different pooling representations, and their combined representations with "fc", under two single tasks and the multiple task settings on Indoor-67. Here the net is fine-tuned on Indoor-67.

and stop it. From Table 7, it can be concluded that all the methods are improved compared with the ones without fine-tuning, and TDP can still outperform other methods.

## 5. Conclusion and Future Work

In this paper, we proposed a new pooling strategy named task-driven pooling (TDP) for image classification. The pooled representations are implicitly learned by maximizing the accuracy of the given classification task and equalizing the similarities between the descriptors and the learned representations. The optimization of TDP can be easily solved by alternative optimization with closed-form solutions for each substep. By considering the CNN feature maps of different scales as the inputs to TDP, we can learn a much better pooled representation for image classification. Moreover, we also proposed a multi-task extension of TDP to further improve the performance. Experiments on three datasets validated the effectiveness of our models, and we achieved the best performance on the Caltech-101 database. Our future work will consider taking other features as the inputs of TDP (mTDP).

**Acknowledgment.** This work was supported by the National Basic Research Program of China (973 Program) Grant 2012CB316302, the Strategic Priority Research Program of the CAS (Grant XDA06040102) and National Natural Science Foundation of China (NSFC) Grant 61403380.

## References


[1] L. Bo, X. Ren, and D. Fox. Multipath sparse coding using hierarchical matching pursuit. In *CVPR*, 2013. 2, 5, 7

[2] Y.-L. Boureau, N. Le Roux, F. Bach, J. Ponce, and Y. LeCun. Ask the locals: multi-way local pooling for image recognition. In *ICCV*, 2011. 3

[3] Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 2010. 3

[4] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. 2, 5

[5] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV*, 2004. 1, 2

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1

[7] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013. 7

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv:1310.1531*, 2013. 7

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008. 6

[10] S. R. Fanello, N. Noceti, C. Ciliberto, G. Metta, and F. Odone. Ask the image: supervised pooling to preserve feature locality. In *CVPR*, 2014. 2, 3

[11] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *CVIU*, 106(1):59–70, 2007. 5

[12] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric p-norm feature pooling for image classification. In *CVPR*, 2011. 3

[13] K. Fukushima and S. Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469, 1982. 3

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

[15] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*. 2014. 7

[16] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv:1406.4729*, 2014. 5, 7

[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014. 6, 8

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2, 3, 6

[19] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 1, 2

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[21] D. Lin, C. Lu, R. Liao, and J. Jia. Learning important spatial pooling regions for scene classification. In *CVPR*, 2014. 6, 7

[22] L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *ICCV*, 2011. 2

[23] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1, 2

[24] N. Murray and F. Perronnin. Generalized max pooling. In *CVPR*, 2014. 2, 3, 4, 8

[25] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *CVPR*, 2006. 5

[26] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 2

[27] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, 2009. 5

[28] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR*, 2014. 7

[29] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999. 3

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 3, 6, 7, 8

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2, 3

[32] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 1, 2, 7

[33] S. Xiang, F. Nie, G. Meng, C. Pan, and C. Zhang. Discriminative least squares regression for multiclass classification and feature selection. *TNNLS*, 23(11):1738–1754, 2012. 4

[34] N. Xie, H. Ling, W. Hu, and X. Zhang. Use bin-ratio information for category and scene classification. In *CVPR*, 2010. 6

[35] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009. 1, 2, 7

[36] G. Ye, D. Liu, I.-H. Jhuo, and S.-F. Chang. Robust late fusion with rank minimization. In *CVPR*, 2012. 6

[37] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014. 3

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

[39] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 7

[40] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010. 2