Feature Combination with Multi-Kernel Learning for Fine-Grained Visual Classification

Anelia Angelova*
Google Inc
anelia@google.com

Alexandru Niculescu-Mizil
NEC Labs America
alex@nec-labs.com

Abstract

This paper addresses the problem of fine-grained recognition in which local, mid-level features are used for classification. We propose to use the Multi-Kernel Learning framework to learn the relative importance of the features and to select optimal features with regards to the classification performance, in a principled way. Our results show improved classification results on common benchmarks for fine-grained classification, as compared to the best prior state-of-the-art methods. The proposed learning-based combination method also improves the concatenation combination approach which has been the standard practice in combining features so far.

1. Introduction

Fine-grained classification [10, 21, 2] focuses on recognition of sub-categories of a base-level category, e.g. recognition of bird species [2] or flower species [16] or dog species [12]. The main challenges of this problem domain are the fine-differences between closely related categories that are categorized as different classes.

A number of approaches have been proposed for fine-grained recognition [24, 22, 16, 17, 3, 4, 2, 20], but only few of them have focused on explicitly making fine distinctions between similar classes [24]. At the same time, identifying fine differences is crucial in fine-grained recognition, since objects of different classes can look very similar at first glance and may only differ in very small local regions (Figure 1). Prior work has utilized local patches for fine-grained classification [22, 24], but such features are selected in ad hoc way, or may not necessarily be optimal with regards to the final classification performance [22]. Other works have focused entirely on manually labeled body parts [24, 10]. Additionally, a common practice in feature combination has been to simply concatenate feature representations coming from semantically different features [15, 22]. Our approach addresses both problems by learning the features corresponding to appropriate local regions, and learning their mutual combination with regards to the classification task at hand. Furthermore, our approach does not require any manual tagging, which is tedious and expensive.

In this paper we address the question: can we refine the feature selection process, so that the features' weights are optimal with respect to the classification performance? We formulate the feature selection process in a Multi-Kernel Learning (MKL) [13] framework. Our framework allows to determine the features' relative importance in a principled way and allows for working with arbitrarily large number of features. Figure 2 shows a schematic of the approach in which we both learn candidate local regions in which the algorithm can focus to identify fine details, as in [22], and learn the optimal combination of feature representations which best solves the task at hand, i.e. by optimizing directly the classification accuracy of the overall task. We refer to our approach as feature combination learning (or feature combination), because we learn the combination of features with regards to improving the classification performance. Our proposed approach is designed to obtain the optimal combination of features for the final classification task. Additionally, the approach allows for selection of the most important features to decrease computational time, for the cases in which the number of features is too large.

Our main contribution is to propose a unifying framework that can extract and learn features which are responsible for fine-grained differences and to learn their combination in an optimal way. Feature combinations based on Multi-Kernel Learning has had various applications, but ours is the first one to show its benefits for making locally fine distinctions, which is the most challenging problem in this domain.

We tested our approach on two commonly used publicly available large-scale datasets: the Caltech-UCSD 200 Birds dataset [21], and the Stanford 120 Dogs dataset [12]. Our approach accomplishes competitive results for fine-grained classification, improving on the state-of-the-art results on these datasets. Our approach is general but at the same time

*This work was done while the first author was at NEC Labs America.

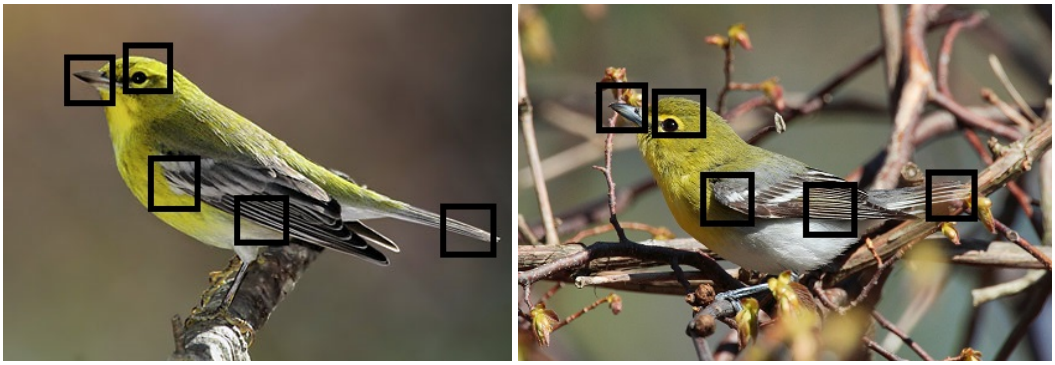

Figure 1. Recognizing a *Pine Warbler* (left) from a *Yellow Vireo* (right), which are quite similar. To discriminate between different species within the same top level category, e.g. birds, we need to focus on the fine details that are present in local regions. This problem is challenging because the recognition system must identify appropriate local regions which are of most discrimination benefit, as well as, computationally combine all the information extracted from each local region, to take the best possible decision. Our paper addresses both problems in a unified framework, without the need for manual annotation of parts.

outperforms a range of methods, varying from segmentation [3, 4], to random forests [23], and including two methods whose ideas have been incorporated in our framework, namely, feature-based selection by template matching [22] and Multi-Kernel Learning [21].

2. Previous work

A variety of fine-grained categorization approaches have been proposed. Yao et al [23] use a random forest for recognition, whereas Farrell et al. [10] propose 3D primitives called 'birdlets', which correspond to bird parts. Other methods improve recognition for fine-grained categorization through segmentation, e.g. [1, 17] or co-segmentation [3, 4]. Some previous works have used fully, manually annotated data or human-in-the-loop [2, 20, 9] to extract features that are useful for categorization. Our approach is on automatically learning and selecting these features with a view towards really large scale fine-grained recognition problems where manual labelling is prohibitive.

Recent work on pose-normalized pooling [24] and template matching [22] are the most relevant to our work. Pose-normalized pooling [24]finds alignment between common parts across different sub-categories by using manual tagging of fiducial body parts. The main differentiator to [24] is that here, both the spatial feature alignment *and* the feature selection are done fully automatically and we do not use manual supervision for identifying body parts, which can be tedious and expensive. Our work builds on the idea to extract local features without manual supervision of Yang et al. [22], who proposed computing a dictionary of 'templates' shared across all classes. This approach however suffers from two major limitations, which our method addresses: Firstly, the features are selected to represent the image data well and not necessarily to discriminate between classes. Secondly, the features computed from the selected dictionary are concatenated together to form a new feature representation, which, as seen in the results, is suboptimal.

Multi-kernel learning [7, 6] approaches have proposed to combine different feature representations, or kernels. They have been successfully applied to computer vision problems [11, 19]. Branson et al [2] applied MKL in the context of fine-grained recognition. More specifically, they applied the MKL algorithm of [19] for a better recognition of birds species using human-in-the loop input. The work of Kumar et al [13] has recently demonstrated that kernel combination for MKL can be re-formulated as a linear classification problem, which allows for handling large number of kernels and can potentially scale up to very large datasets.

3. MKL for feature combination

This section describes how to utilize mid-level features for better classification, by learning their relative importance. We suggest a simpler version for the template dictionary learning method of Yang et al [22] by relaxing some of the constraints. This is described in Section 3.1. These dictionaries can be extracted per arbitrary group of classes, or for all classes together in a global dictionary. One key issue to note here is that, although adding these features in the classification is no doubt useful [22], the features are learned with only 'reconstructive' purposes, i.e. selected to fit the available data well and not with regards to the final classification goal in mind. For example, one selected feature may be common, but only encountered in the background, e.g. a tree branch (see Figure 2). Such a feature may be present in many classes which will make it a good feature, according to the criterion used in [22]. Nevertheless, it will not be a good feature to use in the final classification and thus its relative importance to the classification of birds should be very small, or almost 0. This is why, we propose here to learn the relative weights of the selected

Figure 2. Our paper proposes to learn the optimal combination of visual features (and their representations $\mathbf{w_1}, \mathbf{w_2}, \ldots$, etc.), with regards to the final classification performance. For example, although all features above may be common in birds' images, the left-most two features (of a bird's head or wing) are much more important for fine-grained classification than the right-most two features (possibly of background). We here learn the relative importance of their feature representations $\mathbf{w_i}$ by learning the optimal weight combinations $C_1, C_2, \ldots$, etc.

features (Sections 3.2 and 3.3). In Section 3.2, we first show that *weighted concatenation* of features can be expressed as a linear combination of kernels, which is precisely the MKL formulation. Section 3.3 describes the details of the binary formulation of MKL, proposed by Kumar et al. [13], which we chose here because of its scalability.

3.1. Unsupervised feature learning

The template matching algorithm, of Yang et al. [22] extracts features that can be shared across classes in unsupervised fashion. Since the 'templates' focus on fine-details and are selected in unsupervised manner, we believe they are well suited to the task at hand. We made a few modifications to the algorithm, as described below. The algorithm proceeds by learning a set of regions, or mid-level features (also called templates in [22]) $T_1, ...T_K$ that are common across classes. Introducing auxiliary variables $v_i^I, i = 1 \ldots K$ as indicator variables for each found region in an image I and $l_i^I, i = 1 \ldots K$ as locations where the regions are found, we maximize the following cost function:

$$\max_{\mathbf{T},\mathbf{v},\mathbf{l}} \sum_I \{L_1(\mathbf{T},\mathbf{v},\mathbf{l}) - L_2(\mathbf{T},\mathbf{v},\mathbf{l})\}. \quad (1)$$

As seen, the cost function is composed of two terms. The first one measures how well the templates of the dictionary approximate local neighbourhoods in the images

$$L_1(\mathbf{T},\mathbf{v},\mathbf{l}) = \sum_{i=1}^{K} v_i(1 - \|T_i - R(I, l_i^I)\|) \quad (2)$$

where $R(I, l_i^I)$ denotes the region of image I at location l_i^I. The second term enforces selection of 'diverse' templates, i.e. penalizes selection of templates that are too close:

$$L_2(\mathbf{T},\mathbf{v},\mathbf{l}) = \sum_{i=1}^{K}\sum_{j=1}^{K} v_i^I v_j^I d(l_i^I, l_j^I) \quad (3)$$

where $d(l_i^I, l_j^I)$ is the location penalty: $d(l_i^I, l_j^I) = \infty$ if $\|(l_i^I - l_j^I\| < \lambda$ and 0, otherwise. In this work, we relaxed the constraints on features' locations, allowing features to be found anywhere in the image. We also did not use the co-occurrence criterion of [22]. Additionally, we used a simpler (and also potentially faster representation) based on global pooling of features within the template region. Section 4 has details on the feature representation.

Figures 3 and 4 show examples of learned features for the birds and dogs datasets, respectively. As seen, the method is useful in extracting common features, e.g. a dog's head, but at the same time, some features may be contaminated, e.g. contain a mix of different patterns.

3.2. Feature combination formulation

We here learn the weights of individual blocks of features, each block feature representation corresponding to one mid-level feature. As shown in Figure 2, some mid-level features may be more useful than others, so conceivably the weight of the specific feature representation should depend on the usefulness of the feature for classification.

Let us assume that we have K mid-level features each one described by a feature vector: $\mathbf{w_1}, \mathbf{w_2}, \ldots \mathbf{w_K}$. When concatenating these features we form the new feature representations $\mathbf{F_{concat}}$ which simply concatenates all the features.

$$\mathbf{F_{concat}} = (\mathbf{w_1}, \mathbf{w_2}, \ldots \mathbf{w_K}) \quad (4)$$

We note that this type of feature combination is extremely common in the literature [15, 22], where often very diverse set of features are being concatenated.

Instead, our goal here is to consider combination of features, or *weighted* concatenation, which is of the form:

$$\mathbf{F_{comb}} = (C_1\mathbf{w_1}, C_2\mathbf{w_2}, \ldots C_K\mathbf{w_K}), \quad (5)$$

where the coefficients $C_1, C_2, ...C_K$ are to be learned. Our approach is to select these coefficients in such a way, so that they minimize the expected classification loss. This goal is aligned with the final performance of the system.

As the new feature representation in Equation 5 is going to be used in SVM multi-class framework, we here prove that we can view the same problem as an SVM learning task with additive kernels. More specifically, let $\mathbf{F^1} = (\mathbf{u_1}, \mathbf{u_2}, \ldots \mathbf{u_K})$ and $\mathbf{F^2} = (\mathbf{v_1}, \mathbf{v_2}, \ldots \mathbf{v_K})$ are two different feature representations made from concatenating the feature representations of K different features. Suppose we form the reweighted version by applying fixed weights to each feature block: $\mathbf{F^1_{comb}} = (c_1\mathbf{u_1}, c_2\mathbf{u_2}, \ldots c_K\mathbf{u_K})$ and $\mathbf{F^2_{comb}} = (c_1\mathbf{v_1}, c_2\mathbf{v_2}, \ldots c_K\mathbf{v_K})$. To form a dot product between those features, we obtain:

$$\Phi(\mathbf{F^1_{comb}}, \mathbf{F^2_{comb}}) = \mathbf{F^1_{comb}}.\mathbf{F^2_{comb}} =$$

$(c_1\mathbf{u_1}, c_2\mathbf{u_2}, \ldots c_K\mathbf{u_K}).(c_1\mathbf{v_1}, c_2\mathbf{v_2}, \ldots c_K\mathbf{c_K}) =$

$c_1^2\mathbf{u_1}\mathbf{v_1} + c_2^2\mathbf{u_2}\mathbf{v_2} + \ldots + c_K^2\mathbf{u_K}\mathbf{v_K} =$

$\sum_{i=1}^{K} c_i^2\Phi(\mathbf{u_i}, \mathbf{v_i})$.

That is, the dot product of concatenated features can be represented as linear combination of kernels, in our case these are linear kernels, i.e. $\Phi(\mathbf{u}, \mathbf{v}) = \mathbf{u}.\mathbf{v}$. Thus, we can see that we can represent the combination of features from Equation 5 as weighted sum of kernels, the latter being the MKL framework [11] formulation. In this work we propose to use the above-mentioned property to learn the unknown coefficients C_i (here $C_i = c_i^2$).

In Section 3.3 we will further reduce the additive kernel Multi-Kernel Learning task to a Binary classification Multi-Kernel Learning [14] in which the coefficients C_i of our task can be optimized by increasing the final classification performance.

3.3. Binary classification formulation for MKL

In the previous section we have shown that it is possible to adapt the weights of each feature block by converting the problem to a Multi-Kernel Learning one. We also showed that for linear kernels it is equivalent to the very popular concatenation, but, unlike previous works, here we propose learnable weights for each block. In this section, we further reduce the problem to the binary classification formulation of Multi-Kernel Learning of Kumar et al [14], in order to learn the features' weights. This is done so that the features' relative importance is learned when optimizing the classification performance, rather than other criteria, e.g. minimizing a reconstruction error by approximating the visual appearance of patches that belong to a set of classes, as previously done in [22].

Suppose now we want to optimize $\sum_{i=1}^{K} C_i\Phi_i(u, v)$, where $\Phi_i(u, v) = \Phi(u_i, v_i) = u_i.v_i$ is the projection of the full concatenated feature vector onto the i^{th} feature block. That is, we can view each feature block as individual kernel and we would like to learn the weight C_i in the kernel combination. Kumar et al. transform this problem to a binary classification problem [14]. Similarly, here, for each pair of examples $F = F_{comb}, G = G_{comb}$ we form a new example with representation $Z_{F,G}$ and label $y_{F,G}$:

$$Z_{F,G} = (\Phi_1(F, G), \Phi_2(F, G), \ldots, \Phi_K(F, G)),$$

$y_{F,G} = 1$ if F, G belong to the same class, 0 otherwise. In other words, $Z_{F,G} = (F_1G_1, F_2G_2, \ldots, F_KG_K)$, where F_i and G_i are the corresponding representations on the i^{th} feature block. We aim to learn the weights of the new SVM classification problem, which will be exactly the weights C_i in our original problem.

Clearly, the abovementioned criterion is specifically targeting to optimize for examples of the same class to have similar representations, and examples of different classes to have diverging ones. After the optimization is done, we can revert the problem to the original problem in Equation 5 where the learned weights are being used in the combined representation $\mathbf{F_{comb}}$.

We note that while our MKL formulation allows for non-linear kernels to be used, for computational purposes linear kernels may be preferred. Furthermore, we can use this framework to do a feature selection, as well. If the kernel learning is performed using an L1 regularization rather than an L2, the learned kernel weights will be sparse and many features will be eliminated. In this way, we can select the top features and save time by computing much fewer features. This is also important for scalability, e.g. if for really rich datasets, several features are extracted, it may take a long time to compute them at detection time. We mention this as a practical consideration only and note that we did not need to prune our features since the available datasets are still not sufficiently large.

4. Implementation details

This section describes the implementation details. Our basic feature representation is based on HOG features [8] which are computed at several resolutions to allow for small scale invariance. We further use the locally-constrained coding method to compute the representation of each HOG feature into an 8192-large dictionary as in [15]. This basic representation is combined with a spatial pooling in the whole image or within the boundaries of each feature. For each feature, a global max pooling is done. Each pooling response is treated as individual feature. In our feature combination, we combined the representations (global pooling) of individual mid-level features learned as in Section 3.1, as well as global representation for the whole image. We used the Libsvm [5] implementation of SVM as it allows working with direct kernels. We tuned the regularization parameter of the SVM by using a five fold cross-validation.

In our experiments, all images are resized to 500 pixels at the larger side. When learning the regions for region pooling, we followed the recommendations of [22] and selected image patches which were approximately 1/3 of the image size. Since we worked with fixed 32x32 size patches all images are resized to be of size approximately 100x100. Note that this is done only to find the matching regions for the purposes of speed-up; the final pooling is done in the original image. All individual feature representations are $L2$ normalized prior to being combined, which is a fairly standard practice. This is done in both the cases when MKL is done and when feature concatenation is performed for baseline comparisons. For learning the feature combinations, we use the optimization package PEGASOS of Shalev-Shwartz et al. [18]. There, again, a cross-validation on a small subset of the training data is done.

Figure 3. Patches that belong to two mid-level features, learned from the birds dataset, e.g. of birds' beaks and tails.

Figure 4. Patches that belong to features, learned for dogs. These example features contain parts of dogs' heads and feet.

Method	Accuracy (in%)
Baseline concatenation	14.5
Welinder et al [21]	19.0
Yao et al [23]	19.2
Chai et al [3]	23.3
Deng et al [9]	26.5
Chai et al [4]	26.7
Yang et al [22]	28.2
Angelova et al [1]	30.1
Feature combination (Ours)	**30.5**

Table 1. Classification accuracy on the Caltech/UCSD 200 Birds dataset.

Method	Accuracy (in%)
Baseline concatenation	24.7
Khosla et al [12]	22.0
Chai et al [4]	26.0
Yang et al [22]	36.9
Yang et al [22]	38.0
Feature combination (Ours)	**39.5**

Table 2. Classification accuracy on the Stanford 120 Dogs dataset.

5. Experimental results

We tested our algorithm on benchmark datasets which are commonly used for evaluating novel fine-grained classification algorithms: Caltech-UCSD 200 Birds [21] and Stanford 120 Dogs [12] datasets. We use the standard experimental setups established in prior works that have introduced these datasets [12, 21] and by other prior works that have reported results on these benchmarks. More specifically, we report the mean accuracy of classification, which has been the standard metric for fine-grained classification domain so far [2, 3, 4, 17, 22]. For both datasets we used the bounding box information provided. This choice is also made because previously published results in the literature report their results in these settings and we can compare in those same settings. We also note that we did not compare to experimental results that involve a human-in-the-loop since they utilize additional information [10, 20, 9], and the results would not be comparable.

5.1. Caltech/UCSD 200 Birds dataset

The Caltech-UCSD 200 Birds dataset [21] contains 200 different bird species and consists of 6,033 images and is one of the first benchmarks for fine-grained recognition. Although manual annotations of objects' parts and attributes are available, they are not used in our experiments.

Table 1 summarizes the results of our feature combination method. Compared to the prior state-of-the-art methods, we can see that our method provides an improvement, and outperforms the best known method so far. Our method achieves 30.5% mean average precision. We note here that two of the approaches we compared to in Table 1, namely, the Multi-Kernel Learning approach [21] and the feature-based learning [22], test methods that are similar to the ones we utilize in our framework. Our method outperforms both, which is a very satisfying result.

We compare our method to a baseline algorithm that applies concatenation of features, since this is the most common scenario in prior work. As we can see for fine-grained recognition, where mid-level features are used, the improvement of our feature learning method is quite considerable: 30.5% vs 14.5% for the baseline which concatenates the features without the flexibility of learning weights. The improvement is due to our feature learning method, since this is the only difference between the two approaches. We also note that recent methods have obtained better results with a human-in-the-loop (32.8%) [9], but these results are not comparable since they used additional information. Their comparable results with no manual supervision are 26.5% [9].

5.2. Stanford 120 Dogs dataset

The Stanford 120 Dogs dataset [12] contains 120 different species of dogs and consists of 20,580 images. This is one of the largest fine-grained classification datasets as of this writing.

Table 2 summarizes the results of our feature combination method for the 120 dogs dataset. Our method achieves 39.5% mean average precision compared to 38.0% (36.9%),

26.0% and 22.0% of prior methods. We note that the top competing result of Yang et al [22] 38.0% was obtained when the authors included additional features, based on shape contours. Since our method uses only features corresponding to rectangular patches in the image, we would expect an improved performance of our method with extra shape features. We further compared our results to our baseline method, which does not learn weights. Here the improvement of our feature learning method is also very large: 39.5% vs 24.7% for the baseline.

Discussion. As seen from our results, although the baseline method still allows the SVM to optimize all weights globally and in principle should be able to learn the weights well, this does not happen in practice and the baseline classification rates are rather low. We also note here, that the problem of properly concatenating and normalizing features has been known in the machine learning community, with various normalization methods proposed. Here, we believe the problem is exacerbated by the fact that the mid-level visual features extracted for these specific fine-grained classification problems, and their corresponding representations, may be very sparse and exhibit very strong responses.

6. Conclusions and future work

We presented an approach which learns the feature combination of mid-level features in a principled way using the Multi-Kernel Learning framework of [14]. We showed that it is beneficial to learn the weights, and more specifically to learn them so that the final classification performance is improved. Our approach significantly outperforms its counterpart baseline that does not use learned weights and is competitive to the state-of-the-art methods.

This approach is not specific to the super-category at hand and can be applied to other categories as well, so it will be interesting to see how it compares on man-made categories, such as cars, or airplanes. Another interesting direction is to explore whether MKL can be combined with human-in-the loop approaches, e.g. [20].

Acknowledgements. We thank our team at NEC Labs America. Special thanks to Phil Long for fruitful discussions on the paper.

References

[1] A. Angelova and S. Zhu. Efficient object detection and segmentation for fine-grained recognition. *CVPR*, 2013.

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

[3] Y. Chai, V. Lempitsky, and A. Zisserman. Bicos: A bi-level co-segmentation method for image classification. *ICCV*, 2011.

[4] Y. Chai, E. Rathu, V. Lempitsky, L. V. Gool, and A. Zisserman. Tricos: A tri-level class discriminative co-segmentation method for image classification. *ECCV*, 2012.

[5] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intellligent Systems and Technology*, 2011.

[6] C. Cortes, M. Mohri, and A. Rostamizadeh. Two-stage learning kernel algorithms. *ICML*, 2010.

[7] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel target alignment. *NIPS*, 2001.

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

[9] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. *CVPR*, 2013.

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

[11] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. *ICCV*, 2009.

[12] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for fine-grained image categorization. *Workshop on Fine-Grained Visual Categorization, CVPR*, 2011.

[13] A. Kumar, A. Niculescu-Mizil, K. Kavukcuoglu, and H. Daume. A binary classification framework for two-stage multiple kernel learning. *ICML*, 2012.

[14] N. Kumar, P. Belhumeur, A. Biswas, D. Jacobs, J. Kress, I. Lopez, and J. Soares. Leafsnap: A computer vision system for automatic plant species identification. *ECCV*, 2012.

[15] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: fast feature extraction and svm training. *CVPR*, 2011.

[16] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. *ICVGIP*, 2008.

[17] O. Parkhi, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Cats and dogs. *CVPR*, 2012.

[18] S. Shalev-Shwartz, Y. Singeri, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. *ICML*, 2007.

[19] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multipe kernels for object detection. *ICCV*, 2009.

[20] C. Wah, S. Branson, P. Perona, and S. Belongie. Multiclass recognition and part localization with humans in the loop. *ICCV*, 2011.

[21] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200. *Technical Report CNS-TR-2010-001, California Institute of Technology*, 2010.

[22] S. Yang, L. Bo, J. Wang, and L. Shapiro. Unsupervised template learnign for fine-grained object recognition. *NIPS*, 2012.

[23] B. Yao, A. Khosla, and L. Fei-Fei. Combining randomization and discrimination for fine grained image categorization. *CVPR*, 2011.

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