# Multiple Feature Learning for Action Classification

Benjamin Poole
Computer Science Department, Stanford University, Stanford, CA
poole@cs.stanford.edu

## Abstract

*We review the performance of current state of the art fine-grained image classification algorithms using a variety of features on three datasets. Expanding on these results, we utilize a variety of feature combination techniques to improve performance by incorporating multiple features into classification. We hope to find that multiple kernel learning performs better than unsupervised feature learning techniques on this task. Additionally, we hope to gain a better understanding as to what features are able to capture the fine-grained details of images and how we can best combine these features to achieve the best performance.*

## 1. Introduction

Traditional object classification datasets have focused on objects that are substantially different in their visual characteristics. These datasets generally focus on objects that may be of vastly different sizes, shapes, and colors (e.g. car, plane, chair, person). This focus has led to the development of techniques that are successful at discriminating very different objects, but fail to discriminate similar objects or instances of objects. With the exception of facial recognition, very little work has gone into classifying similar objects such as different types of dogs or cars. These classification tasks rely on very small, fine-grained differences in visual features, such as different ears or tails in dogs. More recent datasets containing humans performing activities and playing instruments has led to new classification techniques, however these techniques tend to rely only on one type of feature (e.g. SIFT or HoG).

In this project, we hope to explore a large set of features for fine-grained image classification, and identify a subset of features that is able to perform well on three datasets: the PASCAL VOC 2010 action classification dataset, the recent People-Playing-Musical-Instrument dataset, and the 15 scene dataset. Initially, we will look at some standard features used in computer vision: SIFT, shape-based templates, contextual features, HoG, Local Binary Patterns (LBP), wavelets, LLC, and color histograms. We intend to evaluate these features using average precision as the metric on a variety of classifiers, including SPM, Multiple-kernel learning, SVMs, and random forests with discriminative decision trees [2]. Our hope is that a certain subset of these features will provide improved performance across all datasets and classifiers, while some features may provide little or no useful information for fine-grained image classification. Furthermore, we hope to beat the state-of-the-art in activity recognition by utilizing a combination of features (instead of just SIFT found in [2]). Through the analysis and review of this variety of techniques, we hope to gain the intuition to develop a new feature set that is able to capture fine-grained information contained in images. Time permitting, we hope to compare these hand-designed discriminative features to semi-supervised feature learning techniques to determine the effectiveness of semi-supervised feature learning at capturing fine-grained discriminative information [1].

## 2. Methods

### 2.1. Image Representation

In our experiments, we use four different types of features. These features were chosen to provide a heterogenous description of image attributes that is able to provide information about shape, color, and texture.

- **Color:** We plan to use a simple color histogram feature. We have yet to determine the exact parameters we will use.
- **Local Binary Pattern (LBP):** This feature has been shown to provide very good performance on texture classification tasks. It represents an image patch by computing a histogram over
- **SIFT:** We use the classic SIFT descriptor with a fixed spacing (6 pixels), and a variety of different scales (8, 12, 16, 24, and 30).
- **HOG:** We use the histogram of oriented gradients descriptor, using the parameter settings and code from Felzenszwalb et. al (2010).

### 2.2. Incorporating Context

The primary dataset we are experimenting with, PASCAL VOC 2010 action classification, includes bounding boxes to denote which person in an image we are classifying. These bounding boxes are somewhat noisy, and often crop out parts of an activity that may be useful. For example, the bounding box for the "ridinghorse" action in Figure 1 crops out a large part of the horse. Many prior studies in image classification have shown that including background information can help to improve performance (e.g. Delaitre et al, 2010). We adopt their foreground-background model here, which consists of two regions:

1. **Foreground:** We rescale the bounding box by $1.5\times$, and resize the image so that the longest edge of the bounding box is 300 pixels. We then use a 3-level spatial pyramid on the features from within the foreground region.

2. **Background:** The background region is computed from the resized image above, and the nwe use a 2-level spatial pyramid on all features within the image.

By enlarging the foreground bounding box, we are able to pick up more fine details closely related to the action. Having a 2-level spatial pyramid for the entire image allows us to represent more of the global context across the image without overfitting detailed features in the background.

Given these two different regions, we will end up with two different kernels. Typically these kernels are averaged, however we will explore more complex techniques for combining this information.

### 2.3. Coding

We used LLC for coding the HOG and SIFT features, with different codebooks of 1024 for each feature type. For color-histogram and LBP we used K-means with 256 codewords (this number will most likely change with mor experimentation).

### 2.4. Spatial Pyramid Matching

For all of our experiments, we compute a set of features from patches sampled uniformly on different sized grids in the foreground and background images. These features are informative, but we need a method to pool over these features to reduce noise and incorporate invariance to scale and translation. As noted above, we use spatial pyramids, which divides an image up into hierarchical regions at multiple scales and computes a histogram over features within each region (Lazebnik et al., 2007). We use the histogram intersection kernel as the features for our final classifier. Note that for each of the 4 different feature types, we have a different kernel that still has to be combined in some way.

### 2.5. Multiple Kernel Learning

Traditional classification using SVMs relies on having a single kernel. To determine which kernel to use generally requires a combination of brute force (trying many different kernels) and careful manual selection (deciding how to weight and combine different kernels into a single kernel). An alternative framework, multiple kernel learning (MKL), allows for automatically learning a linear combination of kernels that performs optimally (Bach et al, 2004). Thus MKL provides a framework for automatically determining how to combine multiple features.

Adopting MKL instead of a traditional SVM will allow us flexibility in two ways: (1) We can weight different types of features differently for each class, (2) we can weight foreground and background elements differently. Being able to alter the weights on the different feature types may be beneficial as certain action classes may be better defined by texture than shape or color. Allowing the foreground to be weighted stronger than the background may also be important in discriminating classes if some classes have very static backgrounds (such as horseback riding), and others have very dynamic backgrounds (such as phoning).

## 3. Experiments

Unfortunatley, I don't yet have results using the PASCAL VOC 2010 Action classificaiton dataset, but for my final project I plan on basing most of my analysis on this dataset. Additionally I have not yet had a change to implement the MKL part, so most of these initial experiments are very basic. However, the framework for feature extraction and classification is in place and thus the remainder of this project should move very rapidly.

The following experiments were on the 15-scene dataset. We did not use the foreground/background model above, and instead just used a 3-level spatial pyramid on the entire image. We extracted SIFT and HOG features only, but will include color histogram and LBP in the future.

### 3.1. Baseline

We first compute the accuracy of each feature individually on all three datasets using an SVM classifier with the histogram intersection kernel. Here we perform a 70/30 split of the dataset, but in the future we will use either 5-fold cross-validation or the designated train/test splits from the PASCAL challenge.

We found that the mean average precision using just SIFT features for the 15 scene categories was 0.82, while the mAP for HOG was 0.67. This tremendous difference in performance is most likely due to the fact that we were using SIFT features at many different scales. When using just one scale of SIFT, performance dropped to 0.71.

### 3.2. Feature Concatenation

The simplest form of combining features is concatenating the resulting histogram intersection kernels for all features and using this to train an linear SVM. We found that when we concatenated the HOG and SIFT features our performance dropped to 0.79. This result could be due to overfitting as we did not alter the slack penalty when retraining. In the feature we will need to do a broader parameter search for each experiment individually. It also may indicate that HOG does not provide much additional informatoin that SIFT does not, and may also be an artifact of properties of the scene dataset.

### 3.3. Feature Averaging

An alternative is to average the histogram intersection kernels from all features. When we did this, we found mAP remained nearly the same at 0.81.

### 3.4. Joint Feature Coding

For coding we used LLC with 4096 codewords, and concatenated the HOG and SIFT features for each spatial region before coding. We also normalized each of the features separately so they were relatively similarly scaled by subtracting off the mean feature for each feature type and dividing by the standard deviation. Using this method, we found that performance dropped to 0.34, indicating that there was either a bug or this technique is very flawed. It may also be that even with normalization, the codebook we learn does not do a good job of representing our feature space.

## 4. Future Work

The focus for the rest of this project will be to get multiple kernel learning working, and reperform all these experiments on the PASCAL dataset. MKL provides a more principled method for combining multiple features and will hopefully improve performance on the someone hacked together methods described above. Furthermore, if MKL does perform well, we believe that it could be incorporated into existing state of the art methods (e.g. Yao et al.'s discriminative randomized forest approach) to improve classificaiton performance even more. Their current state of the art method selects only one feature at each node in a decision tree, but using MKL we could utilize a variety of features to improve performance.

Additionally, the results from the MKL will be compared with features learned from a convolutional deep belief network (CS229 project) in an attempt to see whether combinations of hand-crafted features can beat the state of the art in unsupervised feature learning on an action classificaoitn task.

Furthermore, the experiments being performed wil need to be much more rigorous, as opposed to the very experimental and preliminary results presented above.

## References

[1] J. Ngiam, Z. Chen, P. Koh, and A. Y. Ng. Learning deep energy models. In *International Conference in Machine Learning*, 2011.

[2] B. Yao, A. Khosla, and L. Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. In *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

## 5. Appendix

This project is also my course project for CS229 as well as my rotation project. The computer vision component focuses on manually selecting good features and using them in the MKL framework while the machine learning component is focused on learning features using a convolutional DBN.