

# Multiple Feature Learning for Action Classification

Benjamin Poole

Computer Science Department, Stanford University, Stanford, CA

poole@cs.stanford.edu

## Abstract

*We investigate the performance of various features on an action classification dataset. Utilizing a variety of feature combination techniques, we were able to achieve near state-of-the-art performance using simple classification techniques. We found that multiple kernel learning, stacked feature representations, randomized pooling, and simple cross-validated feature selection all work well in identifying discriminative features for action classification.*

## 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. 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.

In this project, we explore a large set of features for action classification, and identify feature combinations that perform well on the PASCAL VOC 2010 action classification dataset. In particular, we evaluate SIFT, HOG, LBP, and color histogram features with a variety of different parameters. We explore a variety of techniques to combine these features including multiple-kernel learning, stacked representations, and WTA-Hash. We find that all these techniques work reasonably well and help to boost performance over single features.

## 2. Related Work

Most approaches toward recognizing actions in still images have focused on pictorial part-based models (e.g. [6,

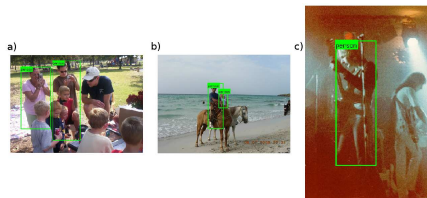


Figure 1. Difficult images from the PASCAL VOC 2010 Action Classification dataset. (a) Additional unlabeled people. (b) Occlusion. (c) Mixture of high and low-quality images.

7, 10]). These approaches introduce a complicated learning procedure, and a much more involved detection and classification stage. In general, they attempt to identify the location and orientation of the humans in a scene, and the overall layout of the human body. To function well, these models must have accurate estimates of location and pose, and thus do not function well in high noise environments. To cope with these problems, more discriminative models have been developed recently that do not require explicit pose estimation. The work by Yao et al. (2010) achieves state-of-the-art performance using randomized forests, but only multi-scale SIFT features. Many of these action-classification techniques ignore other potential features due to the computational constraints involved in incorporating different features into their algorithms. Here we investigate possible feature combinations for action classification to determine whether there exists computationally efficient but discriminative feature sets that can beat the typical standalone features.

## 3. Methods

### 3.1. Image Representation

In our experiments, we use four different types of features. These features were chosen to provide a heterogeneous description of image attributes that is able to provide information about shape, color, and texture. We considered: (1) color histograms, (2) local binary patterns [2], (3) SIFT with fixed spacing at 5 different scales, (4) Histogram of oriented gradients [5].

### 3.2. Incorporating Context

The dataset we were experimenting with 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. To compensate for this problem, we adopt the foreground-background context model of [6]. For each person in an image, we rescale the bounding box by  $1.5\times$ , and use a 3-level spatial pyramid for features within that region. This allows us to pick up finer-grained spatial information in the foreground. For the background, we compute a broad-scale 2-level spatial pyramid over the entire image. Given these two different regions, we end up with two different kernels. Typically these kernels are averaged, however we explore more complex techniques for combining this information.

### 3.3. Coding

We used locally-constrained linear coding (LLC) for coding the HOG, SIFT, and LBP features. LLC has been shown to outperform simple hard-quantization when using SIFT and HOG features [11].

For each feature type, and each region (foreground and background) we learned a different codebook of size 1024. For the color histogram feature we used a codebook with 64 codewords. All codebooks were learned using k-means++, a variant of k-means with a smarter initialization procedure to avoid local optimum [3].

### 3.4. 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 [9].

### 3.5. Combining Kernels

The simplest method for combining multiple features is to take the average across all kernels, weighting each one equally. Recent work on multiple kernel learning (discussed below) has shown that average and product kernels work quite well in practice [8].

To determine which features 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). Alternatively, we can automatically determine the weightings for each kernel for each class using multiple kernel learning (MKL). Multiple kernel learning, allows for automatically

learning a linear combination of kernels that performs optimally [4] for each class.

Adopting MKL instead 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. For this paper we used a variant of MKL called LP- $\beta$  that learns the SVM weights and kernel weights separately. First, an SVM is trained for each kernel independently. Then the weighting of each kernel ( $\beta$ ) is learned based on the output scores from the independent SVMs. The final classifier is then given as the weighted sum of the responses of each classifier. In this sense, the LP- $\beta$  framework resembles boosting, where each of the SVMs represents a weak classifier, and the  $\beta$  coefficients combine them to create a strong classifier. The implementation we used is from [1].

### 3.6. Stacking Features

An alternative approach to MKL is to combine features before coding. Here we compute all our features over the same grid, and simply form a new vector which is the concatenation of all of our features at each point. Given this stacked vector, we perform the same coding and pooling as before. Theoretically the stacking approach has the potential to encode more spatial information about the co-occurrence of features. Thus for action classification, where highly-discriminative fine-grained information determines the class, being able to code these feature combinations may boost performance.

### 3.7. WTA Hash

Instead of pooling over a fixed set of regions (i.e. the hierarchical quadrants defined by SPM), we can pool over random regions using Winner-Take-All (WTA) hashing [12]. This technique randomly selects  $K$  features, and computes the index of the maximum. We repeat this process for a large number of random selections (a.k.a hashes), and we're left with a new feature vector containing a concatenation of the 1-hot index vectors. This method is relatively new, but has been shown to be quite effective and requires extremely small computational cost.

### 3.8. Evaluation

To evaluate our results, we utilize the same metric as the PASCAL competition: average precision. For each feature combination method, we learn a classifier that outputs real-valued confidence estimates for each of the 9 classes. For each class, we then compute the precision-recall curve, and find the area under the curve to get the average precision. To

quantify how well a technique performs across all classes, we compute the mean average precision, which is just the mean of the average precision over the 9 classes.

## 4. Experiments

### 4.1. Dataset

For all the experiments we use the PASCAL VOC 2010 Action classification dataset (<http://pascallin.ecs.soton.ac.uk/challenges/VOC/>). This dataset contains images of people performing any of 9 activities: phoning, playing a musical instrument, reading, riding a bicycle or motorcycle, riding a horse, running, taking a photograph, using a computer, walking. Every image (in both the training and testing set) includes a bounding box around the person of interest. There are approximately 450 total images, containing 600 labeled people. These images are split evenly into a designated training and validation set. There is also a test set, but the labels are not publicly available, thus we treat the validation set as the test set, and split the training set 70/30 to generate a new training/validation set. This dataset presents a number of interesting challenges such as occlusion and poor image-quality as shown in Figure 1.

### 4.2. Individual Features

To determine a baseline to compare to later results, we first looked at each type of feature independently. For each feature (e.g. dense SIFT with scale=16px, HOG, dense SIFT with scale=20px), we first computed a foreground and background dictionary using k-means++. Next we used LLC to encode the features (except for color histogram, where we used hard vector quantization), and pooled using a spatial pyramid with 3 levels for the foreground, and 2 levels for the background. The final kernel is given by computing the histogram intersection for the foreground, and the background, and then averaging these two kernels. We then train an SVM using this kernel. This baseline procedure follows the one in [6].

We found that all the SIFT features performed comparably, with HOG and color histogram doing substantially worse, and LBP barely functioning at all. One of the limitations of the experiments here was that we only utilized one patch size for HOG (16) and one configuration for LBP (16,2). With more scales of HOG or LBP it is possible that one of them could have matched the performance of any of the SIFT features. Another interesting property is that any single feature type does not perform best for all action classes. For example, color histogram was the best for *playinginstrument*, and *LBP* significantly outperformed all other features for *takingphoto*. This is likely due to the scale of LBP matching up with the size of the camera in most *takingphoto* images.

Given that different features perform better for different classes, we can simply take the best feature for each action, and use that to train the classifier for that action. In this way we use different features, but we still do not have to combine them. To select the feature to use, we simply choose the feature that maximizes average precision on the validation set. Using this method we were able to achieve a mAP of 53.4, which is the best of all feature combination methods (Table 2).

### 4.3. Combining Kernels

Next we looked at the accuracy from average and product kernels (Table 2). Curiously, both the average and product kernels perform worse than the maximum feature classifier. In particular, the performance for *takingphoto* drops off significantly. Because we are weighting each feature equally, and we have many scales of SIFT, it appears as though SIFT is dominating the classification choices. In particular the usefulness of LBP seems to have disappeared, and the results for most classes resemble the single SIFT features.

To cope with the unequal distribution of feature types and to learn better weights we used LP- $\beta$ . We found that LP- $\beta$  did not suffer from all the same problems as the simple feature combination methods. In particular, it achieved performance better than the maximum single feature for 3 different classes. However, LP- $\beta$  was still not able to learn the importance of the LBP features for the *takingphoto* action. This was unexpected, but may be due to the very limited amount of positive training instances (only around 30/action). The mean average precision was also extremely competitive at 53.3 (compared to 53.4 for maximum single feature). Comparing this result to the stacked feature vector, we found similar results (a mAP of 53.4), however the stacked technique did not achieve the highest AP in any action category.

category	SIFT20	SIFT24	SIFT30	HOG	Color	LBP
phoning	31.4	29.7	35.5	<b>35.6</b>	26.2	10.9
instrument	35.6	33.2	36.5	29.9	<b>39.3</b>	14.5
reading	<b>42.6</b>	39.6	40.8	32.4	28.9	12.6
ridingbike	37.6	46.6	54.7	30.4	<b>57.8</b>	13.5
ridinghorse	<b>87.6</b>	85.3	84.5	61.3	52.3	14.2
running	64.8	<b>67.7</b>	65.4	44.5	44.2	16.4
takingphoto	9.2	9.3	9.3	10.6	10.7	<b>26.6</b>
usingcomputer	39.1	<b>39.3</b>	36.4	26.6	25.8	10.9
walking	72.1	71.6	<b>72.2</b>	65.2	35.5	22.8
mAP	46.7	46.9	<b>48.3</b>	37.4	35.6	15.8

Table 1. Average precision of individual features for each of the 9 action categories. We removed the first two scales of SIFT features (12 and 16) for brevity. Their mAP were 46.9 and 48.1 respectively.

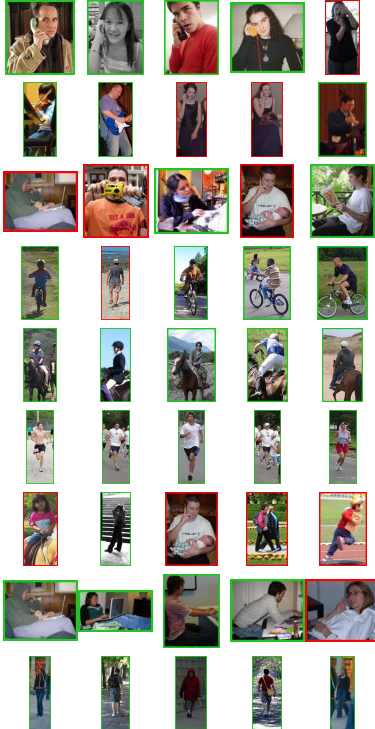


Figure 2. Top 5 objects with the highest score for each of the 9 classes. Each row corresponds to the top 5 objects for a different action. The order of the actions is the same as in Table 1. Correctly classified objects are highlighted in green, while incorrectly classified objects are highlighted in red.

#### 4.4. WTA-Hash

Following the protocol described in the methods section, we computed random hashes maximizing over  $K = 4$  random features from the stacked feature vector. We experimented with varying the size of the hashes, and found that in general more hashes improves performance. Going from 10,000 to 100,000 hashes, the mean average precision went from 46.4 to 50.0. WTA-Hash also had the best performance for *phoning*, *ridinghorse* and *walking*. Furthermore, this technique is extremely fast as we do not have to compute quantizations, LLC, or spatial pyramids. However, WTA underperformed on the difficult *takingphoto* action, most likely because it did not randomly sample enough from the subset of LBP features. Overall, WTA-Hash provided good performance with little computation, and is a promising approach for action classification.

#### 4.5. Qualitative Evaluation

To gain a better understanding of the flaws in our approach, we performed more analyses on the results of the maximum single feature technique. In Figure 2, we show the top 5 objects for each of the 9 classes. For the phoning, running, walking and riding a horse classes we do quite

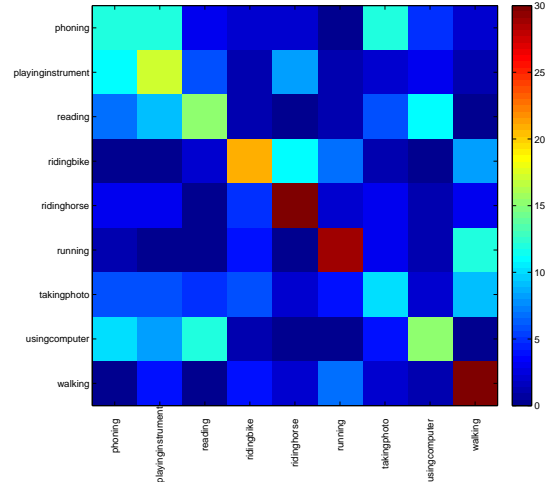


Figure 3. Top 5 objects with the highest score for each of the 9 classes. Each row corresponds to the top 5 objects for a different action. The order of the actions is the same as in Table 1. Correctly classified objects are highlighted in green, while incorrectly classified objects are highlighted in red.

well. However for many of the other classes we make a few mistakes even in the top 5 objects. For example, two women on phones are confused for playing an instrument, probably due to their stance and attire. The classifier seems to pick out more of the stance of the person and less of the object that is involved with the action.

To get a slightly more quantitative look at these results, we computed the top 50 objects for each action, and then looked at the distribution of true labels for these objects. This yields a confusion matrix in Figure 3, where each row represents the distribution objects that are classified as belonging in that row. Here we see that *ridinghorse*, *running*, and *walking* are mostly along the diagonal. But *phoning* is often confused to be using a computer or playing an instrument. These results again seem to reinforce the notion that we're picking out more pose-based features and less object-based features.

#### 4.6. Comparison to state-of-the-art

Although the results for performance on the validation set were not released on the PASCAL VOC 2010 website, they do release the testing performance. If we assume the validation set and testing set are similarly distributed, then it is likely that our performance on the testing set would only increase as we would have more training data. However, this assumption may not be true thus these comparisons should be taken with quite a few grains of salt. The results we are comparing to can be found here:

<http://pascal.in.ecs.soton.ac.uk/challenges/VOC/voc2010/results/index.html>

In 6 out of 9 categories we do substantially worse than state-of-the-art for all methods. However for *reading*, our

best performance is 50.5, while the competition’s best is 35.9. Our results for *ridinghorse* are also competitive, we had 89.4 compared to 89.7. For walking we again beat the competition with 74.2 compared to 72.5. Thus despite the naive and simple techniques we use, along with the relatively small variety of features, we are still able to achieve reasonable results. Most of the models in the competition utilized either pose estimates or part-based models. These structured models can tremendously boost performance as shown in [7, 10, 13].

## 5. Conclusions

We explored a variety of feature combination techniques, and found that we could achieve good performance on action classification without explicitly modeling poses or parts. By incorporating a variety of different features, we were able to boost performance on an action classification task with little additional computational cost. Furthermore, we believe that utilizing randomization techniques (such as WTA-Hash) can help to automatically identify regions to pool, and reduce the computational overhead of quantization and classical spatial pooling.

In the future, action classification models should incorporate multiple features to help boost performance. Whether the techniques described in this paper can be applied depend on the particular application, but we believe that incorporating multiple feature types should boost performance on a variety of datasets. Future work should address this question in a concrete setting by evaluating whether incorporating any of these techniques into existing single-feature type architecture (such as in [13]) improves performance.

Table 2. Average precision (%) for all techniques for each of the 9 action categories. Bolded entries indicate the largest value in the row.

category	MaxFeat	Avg	Prod	LP- $\beta$	Stacked	WTA10k	WTA50k	WTA100k
phoning	35.6	33.2	32.6	28.7	37.6	37.6	<b>41.6</b>	39.3
instrument	<b>46.2</b>	38.7	32.8	45.0	45.2	34.9	34.3	36.5
reading	43.4	38.7	32.3	<b>50.5</b>	44.0	42.4	41.9	36.9
ridingbike	54.7	43.6	43.4	<b>61.0</b>	54.0	41.8	60.1	60.9
ridinghorse	87.5	86.9	83.7	88.0	87.8	79.9	85.9	<b>89.4</b>
running	67.7	62.9	58.9	<b>70.9</b>	67.6	61.9	63.1	64.9
takingphoto	<b>26.6</b>	9.2	18.1	17.4	26.6	9.4	9.2	9.2
usingcomputer	<b>46.9</b>	42.6	35.0	44.6	45.1	36.5	36.1	39.1
walking	72.2	73.1	61.9	73.5	72.4	72.7	74.0	<b>74.2</b>
mAP	<b>53.4</b>	47.7	44.3	53.3	<b>53.4</b>	46.4	49.6	50.0

## References

- [1] <http://www.vision.ee.ethz.ch/~pgehler/projects/iccv09/>.
- [2] T. Ahonen, A. Hadid, and M. Pietikainen. Face Recognition with Local Binary Patterns. pages 469–481. 2004.
- [3] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA ’07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.
- [4] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, ICML ’04, pages 6–, New York, NY, USA, 2004. ACM.
- [5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.
- [6] V. Delaitre, I. Laptev, and J. Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2010. updated version, available at <http://www.di.ens.fr/willow/research/stillactions/>.
- [7] V. Delaitre, J. Sivic, and I. Laptev. Learning person-object interactions for action recognition in still images. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.
- [8] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.
- [9] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR ’06, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.
- [10] S. Maji, L. Bourdev, and J. Malik. Action recognition from a distributed representation of pose and appearance. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [11] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010.
- [12] J. Yagnik, D. Strelow, D. A. Ross, and R.-s. Lin. The power of comparative reasoning. *csutorontoca*, 2011.
- [13] 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.

## 6. Appendix

This project is also my course project for CS231A as well as my rotation project with Prof. Li. The computer vision component focuses on manually selecting good features and developing the baseline classification algorithms. The machine learning component is focused on utilizing MKL and WTA-Hash and comparing to the manually chosen features.