Scene Classification and Image Retrieval Using SPM and LLC

Nikil Viswanathan and Evan Rosen
Department of Computer Science
Stanford University
{nikil,erosen}@cs.stanford.edu

Abstract

We explored the problem of image classification using a family of methods based on bag of words *features. Using the Spatial Pyramid Matching technique of Lazebnik et al. [2] we were able to add scene structure information to the standard bag of words model. As an extension of this model, Wang et al. [3] propose a novel way to map local image descriptors to visual codewords in the Locality-Constrained Linear Coding. Finally, we experimented with adapting the Spatial Pyramid representation to a scalable image retrieval system using Locality Sensitive Hashing.*

1. Introduction

We compared the performance of Structural Pyramid Matching and Locality-constrained Linear Coding over a suite of values for tunable parameters. Brainstorming about the real world applications of scene classifications led to the implementation of an LSH technique for image retrieval. Ultimately, though we saw promising performance from our methods, memory limitations capped our potential performance even after space complexity optimizations.

2. Scene Classification

The task of image classification plays a central role in both the human visual experience and a variety of computer vision tasks. The class of an image can serve as a key piece of contextual information, useful in resolving the manifold ambiguities which arise in visual intelligence tasks. For example, knowing the class of an image would give excellent evidence for the type of objects we might expect to detect, or for the general 3-D composition. We would even expect different image classes to be amenable to different segmentation techniques. (For example, the edges on a desk should indicate object boundaries, however the edges within tree foliage should not.) However, while this tight relationship between image classification and other vision tasks indicates that it is an important problem, it also points out one of its fundamental difficulties: there is no obvious representation to use for image classification because the relevant signals seem to come from so many different sources.

In the same way that segmentation and object detection are inherently tied, by virtue of their mutual definition (good segments tend to be objects and objects tend to be good segmentations), image classes can be thought of in terms of object detections, depth reconstructions and segmentations. What do we mean when we say that a particular image is of a certain class other than that it contains certain objects classes, possess a certain characteristic segmentation, and corresponds to a view of a certain type of 3-D environment? Thus, while solving the task of image classification would greatly improve many other vision tasks, it comes with the drawback that an *image class* is inherently an open ended notion, whose characteristic features might be best parameterized by a variety of image features. And so a large part of the problem is finding the appropriate representation for an image which can effectively capture the vague notion of an image class. The *bag of words* model has been one of the most successful answers to this question.

The general idea of *bag of words* models, is to find a set of local descriptors called visual words, which generalize across images, and then to represent an image as a distribution over these words. Contrary to the standard notion of a linguistic word, visual words do not naturally fall into discrete well defined categories. So, while any written use of a linguistic word will be immediately recognizable as an instance of a particular type of word, there exist no predefined analogs for visual descriptors. Local image descriptors may fall anywhere in a high-dimensional feature space and we must first find the right way to partition this space, before we can describe a image as a distribution over a finite set of visual words. This step is standardly done with a simple clustering algorithm like k-means, whose cluster centers compose the codebook or dictionary. Then, given a new image, we can map each real-valued local descriptor to a specific visual word in that codebook, and we can begin to treat images as collections over discrete elements.

The metaphor of visual words also struggles with the fact that there is no obvious way to decompose an image into a set of local image descriptors. One approach is to sample these descriptors at regular intervals. However, this can miss many of the discriminative features for image classification. For example we might expect corners to be very discriminative between city and forest scenes. However, if we simply sample local features such as SIFT on a regular grid, we might miss many of these corners. A natural solution to this problem would be to use a keypoint detector like the Harris corner detector. While this would preform very well for city versus forest classification, it would might not generalize well to natural scenes such as ocean or plains in which such a detector might rarely fire. In general the absence of keypoints can be thought of as a discriminating feature for certain scene classes.

Having constructed the analogy to linguistic words, many existing methods in natural language processing can then be applied to images. For example, the set of algorithms known as topics models, used in text summarization and clustering, can be adapted in a straight forward way to image summarization and clustering. These models represent an image as a mixture of visual topics, which are themselves just distributions over visual words. We might think of an image as mixture of sky, forest and city, where each such region is characterized by its own distribution over visual words.

Fei-Fei and Perona [1] use an approach of this sort for natural scene classification. Using a novel, supervised variant of Latent Dirichlet Allocation (LDA), they posit a generative story by which a class label ultimately yields a set of visual words. Each image begins by drawing a distribution over topics according to its category. Then, the topic associated with each local descriptor is then drawn from this image specific distribution over topics. Finally, the particular visual word associated with that local descriptor, is then chosen from a topic-specific distribution over visual words for the topic to which that visual word has been allocated. At test time, a particular set of visual words in a test image can then be used to infer the maximum likelihood class label.

3. Algorithm

We implemented various combinations of the Spatial Pyramid Matching technique of Lazebnik et al. [2] and the Locality-Constrained Linear Coding method of Wang et al. [3].

3.1. Spatial Pyramid Matching

In general, the success of pure bag of words models is somewhat surprising, for they operate without any notion of the spatial layout of visual words. This means that the presence of a codeword which tends to represent grass would be treated as identical evidence for a scene class when it occurs at the bottom of an image and at the top. In an attempt to reintroduce such information, Lazebnik et al. [2] proposed the Spatial Pyramid Matching (SPM) technique. The basic intuition of SPM is that we can use a collection of spatially constrained "bags" to capture characteristic locations of certain visual words. Specifically, a pyramid of successively finer grids is placed on top of the local feature descriptors. All of the visual words which fall within each cell of this pyramid are then pooled together to form the canonical bag or histogram. The visual words in this model are simply the 128-dimensional SIFT feature vectors which have been computed on a regular 16 pixel grid. As we noted above, the visual words approach requires that we represent the continuous space of local feature descriptors using a finite set of descriptor classes. To find these classes k-means is run on the SIFT features from all of the training images to yield around 200 codewords.

Then, after mapping each of the local descriptors to their nearest codebook entry, the pyramid binning scheme described above can be applied to create a collection of spatially constrained histograms. Finally, all of the histograms for a given image are concatenated to form a single feature vector. At this point, the SPM model reduces to a standard multi-class classification problem. To do this, Lazebnik et al. use an SVM with a non-linear, histogram intersection kernel [2] in which the similarity between two feature vectors is just the minimum bin value. Thus two images are considered very similar if they allocate their probability to the same visual code words to similar degrees.

3.2. Locality-Constrained Linear Coding

Building off of the state of the art Spatial Pyramid Matching [2], Wang et al. developed a new technique for mapping a feature descriptor to a codeword group in a sparse and local fashion. In contrast to the Vector Quantization method of assigning each descriptor to the nearest codeword, LLC extends the notions proposed in ScSPM [4] of a sparse nonlinear coding. The authors of ScSPM noticed that the codes tended to be local relative to the descriptor many times and suggested a method of designing local codes for each descriptor. They displayed a theoretical result which claimed that locality was more essential than sparsity when solving a nonlinear SVM optimization problem. The LLC method again extends this idea and creates a local linear approximation to the descriptor by projecting it to a local coordinate space and using a linear kernel is able to achieve high performance with a very fast implementation. This locality constraint also guarantees sparsity through the nature of the constraint and allows for scalability by reducing the nonlinear kernel SVMs running time from O(n3) in which n is the number of support vectors, to O(M + K) with M clustered codebook entries and K nearest neighbors. In order to describe the feature in terms of local codewords, the LLC method performs a K-nearest-neighbor search to find the closest codewords and then solved s constrained least square optimization problem to discover coefficients.

4. Code

- runTests.m - Our main control file for the scene classifier, which contains all of the parameter options.
- BuildPyramid.m - Builds the pyramid feature vectors
- lshRetrieve.m Builds and test the LSH system.
- nearestNeighbor.m - Approximates the quality of the image retrieval.
- computeClosestNeighbor3.m - Contains the evaluation algorithm for selecting the closest neighbor from the top 3 neighbors.

5. Extensions

5.1. Optimizations for speed

Calculating the codeword dictionary, building the spatial pyramid, and running the SVM were memory and compute intensive processes for our machines. Our evaluation systems had 1-2 gigabytes of RAM so we implemented several optimizations to squeeze the last bit of performance from our models in training. To make k-means tractable, we learned a codebook for each class separately and then combined these into a single global codebook. Specifically, we clustered the descriptors from a random sample of 50 images in each class into 100 clusters, yielding a codebook of size 1500 for the outdoor scene category dataset and 1200 for the PPMI dataset. Having removed this bottleneck, it became apparent that the next limiting step lay in compiling the spatial pyramids.

Due to a lack of contiguous memory to store the pyramid array, we eliminated holding the train array in memory during the calculation of the test array by saving it out to file and hydrating the structure when needed. After increasing the size of the training set, we discovered that even compiling one pyramid at a time in the inner function before saving it to file was crashing due to a lack of contiguous variable space (our computers had < 1.2 GB of continuous ram space) and we attempted the same trick of saving each individual image pyramid to file and then assembling the entire pyramid outside of the inner function; this allowed for only slightly more memory intensive computations.

5.2. Image Retrieval

Closely related to the task of image classification is the task of image retrieval. However, whereas image classification usually exists in a supervised framework with a closed set of classes, image retrieval can be applied without supervision, even without any explicit prior knowledge about the classes. Specifically, we explore the case in which given a query image we would like to find similar images. One approach in these situations is to find some way to embed images in a space which makes good matches nearby. We were interested in testing whether the combination of the spatial pyramid match representation with the histogram intersection kernel could be used in this way. We then consider the nearest neighbors to a query image as the retrieval results. While were able to visually evaluate our work by simply inspecting the nearest neighbors by hand, we also used the class labels to construct a coarse automatic evaluation metric. The rate at which the nearest neighbor class matched the query image class are reported in row 1 of Table 1.

Database Size	10	50	75	100
NN with Hist. Int.	66	62	-	-
3-NN with Hist. Int.	56	61	-	-
LSH $L-1$ Accuracy	56	56	59	59
LSH $L-2$ Accuracy	48	48	45	45

Figure 1. KNN and LSH classification results versus number of images in database

Interestingly these evaluation results are simply the classification accuracy for the one of the simplest algorithms, nearest-neighbor interpolation. On this basis we explored the low hanging fruit of image classification by creating a hierarchical 3-nearest neighbor algorithm. Despite the fact that such an algorithm will almost always perform worse than a more complex classification model such as an SVM, it can serve as a direct way to intuitively evaluate an image representation and distance metric. By inspecting pairs of images which are considered close, we can gain an intuition for the types of image attributes which our model is rightly or wrongly drawing attention to. Consider the results in Figure 2. We can see that the location of the houses in (a) and (b) are very similar, which confirms our intuitions about the value of using spatial. Similarly, the fact that both houses have fences in them reminds us that the gradient based nature of SIFT does a very good job at encoding detailed patterns. In images (c) and (d) we can see that the algorithm successfully picked out a similar scene class. However, if we note the drastic difference in the sky composition, we can confirm that our representation must not be assigning great importance to such attributes. While this is acceptable for road classification, we can see why SIFT descriptors might not be ideal for scenes characterized by relatively smooth, continuous regions, like sky and clouds.

While finding a good representation and distance metric may work well in small cases, it is not enough for systems like Google image search, which operate over vast

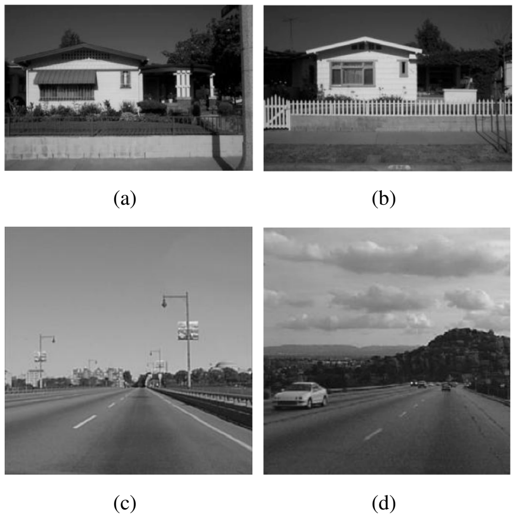

Figure 2. NN matches

databases with billions of images. In these situations, a straight forward comparison of the query image with the entire database becomes intractable. A common way to cope with this problem is to use a technique known as *locality sensitive hashing* (LSH). In the LSH framework, a composition of hash functions is constructed such that the probability that two images hash to the same bin is proportionate to the similarity between those two images by some metric.

To use such a framework, we simply hash all of the images in our database and create an inverted index from the bin indices to the images. Given a query image we can then just compute the hash value and find the appropriate postings list in our inverted index. The main difficulty in LSH applications is finding a construction of hash functions which approximate a distance metric. Though we would ideally find an LSH family to encode the histogram intersection kernel, we can at least evaluate the spatial pyramid representation with simple distance metrics like the L_1 and L_2 norm. Using a matlab implementation of L_1 and L_2 approximation families by Indyk et al. we were able to get surprisingly close to the optimal behavior of our exact nearest neighbor interpolation algorithm. An example results can be seen in Figure 3.

6. Results

Unless otherwise stated, the results shown were run with LLC on 30 training and 20 testing examples, a 1000 codeword dictionary, 3 spatial pyramid levels, a k closest codewords value of 5, max patch pooling, sum scale pooling, without the histogram interesction kernel, and on the 15 class scene categories dataset.

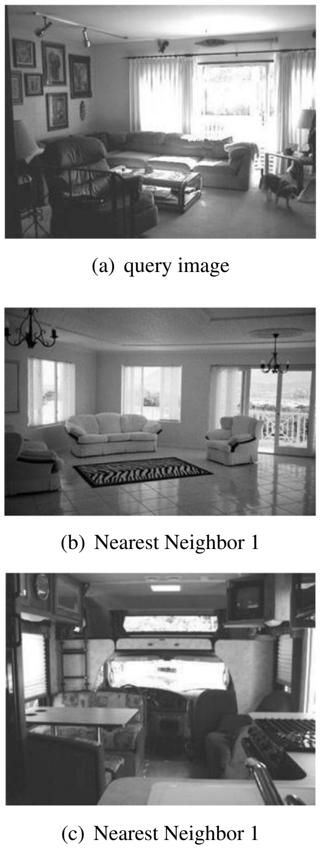

Figure 3. LSH example retreival

6.1. Training Set Size

After optimizing the codeword dictionary generation and the pyramid compilation process we were able to boost up the codebook size to 1000 and run up to 30 training and 20 test examples. Observing the almost linear increase in performance (Table 4) as the size of the training set grows, we postulate that with a more powerful machine we could see better performance by still increasing the training set size. The 30 training examples and 20 test examples completely max out the memory on our machines and while testing we discovered that using this test to train ratio gave us the best performance.

Training Set Size	10	20	30
Test Set Size	20	20	20
Acc (%)	56.7	65.3	77.0

Figure 4. Effect of training set size on accuracy.

6.2. Pyramid Levels

Varying the number of spatial pyramid levels (Table 5) only affected performance slightly with a convergence of the best performance around a 3 level pyramid structure. The variance may be due to the small training set size and the small test set size. With PPMI we observed that 4 spatial levels had the best performance and we postulate that since this task of activity classification is slightly different from scene classification, the increased number of spatial granularities allows the algorithm to hone in to particular regions of interest which works better for this problem since we are only concerned with a particular region of interest (the human and the instrument) instead of the entire image as in scene classification.

Pyr. Levels	1	2	3	4	5
Acc (%)	66	73	77	75	Out of Memory

Figure 5. Effect of number of pyramid levels on accuracy.

6.3. Varying k

Varying the number of the nearest neighbor codewords that a feature descriptor is a linear combination of resulted in minor changes in performance (Table 6) but did not show a consistent trend. The twin peaks at values of 5 and 10 for k, and especially the value at 10, seem to be an artifact of the small training and test set sizes as we would expect to see a steady increase in performance towards one value of k and a decrease in performance afterwards.

k	1	3	5	7	10	12
Acc (%)	71.6	75.7	77.0	75.3	76.3	75.0

Figure 6. Effect of number of nearest neighbor (k) on accuracy.

6.4. SVM Slack

Surprisingly modifying the SVM slack had no effect on the classification performance of our model (Table 7). Training and testing on such a small dataset would suggest that model could be overfitting to the data, which would have been improved by introducing and using a larger slack variable and aggravated by reducing and eliminating the slack variable. We conclude that we have enough training data to create a model that accurately learns the structure of the entire scene data corpus and doesn't overfit to the training set.

SVM Slack	None	1	10
Acc (%)	74.0	74.0	74.0

Figure 7. Effect of SVM slack on accuracy. Using 4 spatial pyramid levels and max patch and max spatial level pooling.

6.5. Pooling

We experimented splitting up the feature descriptor spatial pooling into two stages and at each step implemented sum pooling and max pooling. In the step of compiling the feature descriptor within a given region to form the bins on the lowest level spatial pyramid, we implemented two approaches: 1) simply creating a weighted histogram of the nearest codeword neighbors values of the feature descriptors contained in the patch and 2) choosing the maximum weight seen for each codeword over all of the features in that patch. Secondly, in the process of combining bins of a lower spatial pyramid level to construct an upper level bin we implemented a max value selection technique in addition to a summing approach. We noticed LLC achieved the best performance when using max pooling in both cases.

6.6. Histogram Intersection Kernel

The histogram interesction kernel when applied to different sets of parameter values, often decreased the performance of LLC. In our optimal parameter setting, using a linear kernel vs the histogram intersection kernel actually had no effect upon the classification accuracy.

6.7. People Playing Musical Instruments

We also experimented on the People Playing Musical Instruments (PPMI) dataset [5] which tests the robustness of our classification systems in a very different way. Intuitively this task requires greater attention to detail and may place different demands on the bag of words model. For example, we might expect the important discriminative visual words to be relatively sparse in classifying musical instruments, whereas scene classification might rely on the visual words which occur frequently over uniform textured regions like grass or forest. This suggests that the PPMI dataset might benefit from a keypoint-based local descriptor sampling approach, as we would expect instruments to be relatively rich in keypoints in comparison to background regions.

Model	Codewords	MultiSIFT	Pyr. Levels	Test Acc.
SPM	200	True	4	0.14
LLC	1000	True	4	0.28
LLC	1000	True	3	0.35

Figure 8. Results on the PPMI dataset on 100 train and 100 test images

7. Conclusion

Our overall results are shown in Figure 9. Overall we saw promising performance trends and believe that with additional computational resources we can continue to improve accuracy. Moreover, we also demonstrated that the spatial pyramid feature representation even without the histogram intersection kernel can provide a useful basis for scalable image retrieval. Exciting directions of future research include the both the enhancement of the SPM feature representation for image retrieval and possibly the development of an LSH approximation for the histogram intersection kernel.

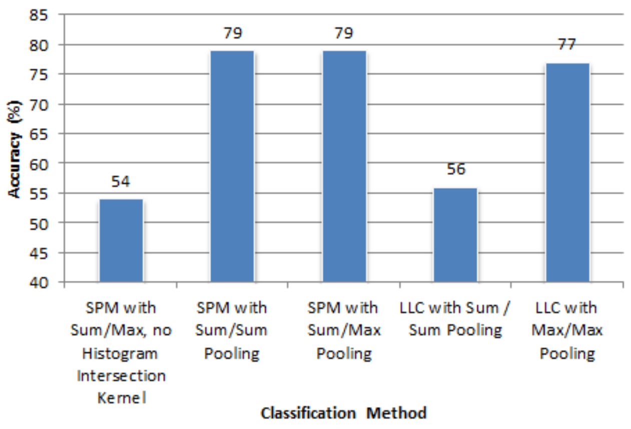

Figure 9. Summary of results.

References

[1] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531. Ieee, 2005.

[2] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. Ieee, 2006.

[3] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. 2010.

[4] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. 2009.

[5] B. Yao and L. Fei-Fei. Grouplet: A structured image representation for recognizing human and object interactions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, USA, June 2010.