Learning Hierarchical Bag of Words using Naive Bayes Clustering

Siddhartha Chandra[1], Shailesh Kumar[2], and C. V. Jawahar[3]
[1]siddhartha_c@students.iiit.ac.in, [2]shkumar@google.com, [3]jawahar@iiit.ac.in

[1,3]CVIT, IIIT Hyderabad, [2]Google, Hyderabad

Abstract. Image analysis tasks such as classification, clustering, detection, and retrieval are only as good as the feature representation of the images they use. Much research in computer vision is focused on finding *better* or *semantically richer* image representations. Bag of visual Words (BoW) is a representation that has emerged as an effective one for a variety of computer vision tasks. BoW methods traditionally use low level features. We have devised a strategy to use these low level features to create "higher level" features by making use of the spatial context in images. In this paper, we propose a novel *hierarchical feature learning framework* that uses a *Naive Bayes Clustering* algorithm to convert a 2-D symbolic image at one level to a 2-D symbolic image at the next level with richer features. On two popular datasets, Pascal VOC 2007 and Caltech 101, we empirically show that classification accuracy obtained from the hierarchical features computed using our approach is significantly higher than the traditional SIFT based BoW representation of images even though our image representations are more compact.

1 Introduction

Over the years, research in computer vision has tried to narrow down the gap between raw image pixels and what humans see when they look at the image. The efforts to do so can be very broadly categorized into two classes. The first class of methods uses a robust representation based on relatively low level features (e.g. SIFT based Bag of Words (BoW) representations [1, 2]). BoW representations have received widespread success in a variety of computer vision tasks such as image classification and object detection [1–3] owing to their invariance to scale, spatial and rotational distortions. These methods also use domain knowledge. Over the years, much research has gone into improving the performance of models that employ BoW representations. Non-linear SVMs, specialized kernels of different kinds [4–7], and spatial pyramids [8] have all contributed to the success of these representations. Most of these approaches tend to increase the overall image representation size. While these methods have been successful in capturing diversity in image patches at low levels, it is hard to incorporate them into a hierarchical feature learning frameworks naturally as there is no systematic way to create higher level symbols from combinations of lower level discrete symbols.

The second class of methods continues to *enrich* low level features by using *hierarchical feature learning paradigms*. Most hierarchical feature learning approaches such as Deep Belief Networks [9], Convolutional Neural Networks [10], Convolutional Deep Belief Networks [11] and other hierarchical models [12] use simple building blocks such as a logistic function to learn complex overall models with many parameters to tune. Ideas like layered incremental training have made the training of these models practical. Further they can model translation invariance well using max pooling. The limitation in these models is that they need real-valued inputs at each layer in the hierarchy and hence the traditional BoW approaches that generate a large number of discrete symbols at lower levels cannot be naturally incorporated into such hierarchical feature learning frameworks. While the deep learning methods have been known to work well for a variety of simpler datasets such as ILSVRC 2010, ImageNet and Hollywood 2, they haven't enjoyed much success on more challenging datasets such as PASCAL [13].

Research into bridging the gap between these two directions exists. Hyperfeatures [14] exploit the spatial co-occurrence statistics at scales larger than their local input patches by aggregating local descriptors using methods such as GMM and LDA. This paper is a step forward in this direction and tries to combine the strengths of hierarchical feature learning and BoW learning paradigms. We propose a generic framework for building *discrete feature hierarchies* in an unsupervised fashion starting from any first level symbol image (e.g. dense SIFT visual words). The framework has two parts: (a) a novel *Naive Bayes Clustering* algorithm that clusters symbolic image patches using EM like updates to maximize the log likelihood of the data in terms of a mixture of naive Bayes discrete multi-variate distributions, and (b) a maximum pooling on neighboring patches using the posterior probabilities of clusters in data points to reduce the image size at the next level. Evaluations on Caltech 101 and Pascal VOC 2007 indicate our compact, meaningful representations outperform the traditional BoW and deep learnt representations.

2 Background

In this section we briefly summarize the two prominent directions in computer vision which endeavour to represent the visual world through features which are invariant to various external parameters. However, while the BoW approaches can improve by further exploiting the information content in the spatial layout of images, the deep learning methods can enhance their utility by overcoming the training and architectural complexity for learning large scale computer vision systems. We intend to learn from these two directions and take an approach that combines their powers to achieve superior representation.

2.1 Beyond Bag of Words

Visual BoW draws its inspiration from the analogous BoW models for document representation that ignore the order of words. Traditional image classification involves computing local features at interest points in an image and pooling these local features to give a global image representation. BoW essentially quantizes each local feature into one of the visual words using a codebook and then represents each image as a histogram of visual words. Computing the codebook involves identifying interesting local patches in an image, extracting features or keypoints such as SIFT from these local patches and finally clustering (usually using K-means) to group key points from the training images into clusters; the center of each cluster corresponds to a different visual word. Finally each SIFT vector is quantized by assigning it the label of the nearest cluster center. We represent each image as a histogram of the visual words, called the BoW representation.

BoW represents an image using the distribution of visual word occurrences. In doing so, it converts images of different sizes into fixed length representations. This is especially convenient for the classification task that need fixed dimensional inputs. However, BoW relies only on the appearance of the visual words and ignores their spatial layout. This characteristic imparts invariance to scale, translation and deformation, at the cost of discriminative power especially when the spatial layout is important.

There have been many recent attempts to overcome the limitations of BoW [15]. These include part generative models like [16] and frameworks that use geometric correspondence search [17]. These work well but are computationally expensive. BoW can be enhanced [18] by extending the codebook to include *doublets* which are pair-wise relations between features that lie in the same local neighbourhood. Spatial pyramids [8] was a major breakthrough in this direction; it incorporates spatial information by computing BoW representations for different image regions at different scales and concatenating these representations and finally uses a pyramid matching kernel [19] for classification. Almost everything in the book - from kernels [4–7] to sparsity [20] to local codes [6] has been attempted to enhance the power of these low level representations [21].

All these indicate the need of raising the semantic depth of the low level features discovered through the BoW process. Bringing context of neighboring features to define "higher level" features is clearly recognized as the next natural step here. As described in section 1, hyperfeatures [14] were devised especially to fulfil this need. In this paper, we continue to explore this middle ground.

2.2 Deep Learning

Deep learning networks [9, 11] and convolutional networks [10] represent an orthogonal school of thought. These are driven by the idea that good internal representations are hierarchical and can be learned directly from the data. These networks have hierarchical layers (also called *feature maps*) stacked together; each feature map learns artifacts in the image by assembling smaller artifacts learnt by the preceding feature maps. Pixels are assembled into edges, edges into object parts, object parts into objects, and objects into a scene; deep learning thus exploits the spatial information in the images. These levels represent the feature hierarchy.

Convolutional networks typically work on *overlapping* patches at each level (the overlap takes care of small translations) and summarize the features learnt in a neighbourhood by a pooling method. Popular pooling methods are (a) average pooling where the features computed over a region (called a cell) are averaged to give the feature representation of the cell and (b) max-pooling where the feature representation of the cell is the maximum of the features in the cell. Convolutional networks usually alternate between feature maps and pooling layers to achieve invariance to small translations and distortions.

Deep learning gives robust image representations. However *insufficient depth can hurt.* Also, training deep networks involves making many design decisions, huge training set, is computationally challenging and most of the feasible training algorithms are mostly approximations of the actual objective. (In fact, attempts at training deep networks had failed before [9].)

There have been a few attempts to bridge the gap between the two schools by taking the middle ground and developing frameworks that exploit the advantages of both [22]. In this paper, we intend to take a leap forward in this direction. Our approach involves deep / hierarchical learning of higher level discrete symbols from lower level discrete symbols (for instance BoW visual words) that lie in the same spatial neighbourhood. Our approach is different from traditional deep learning in that we work with symbols / visual words and not real valued features. A novel Naive Bayes Clustering method allows us to cluster combinations of low level symbols.

3 Naive Bayes Clustering

In order to build hierarchy of discrete features to compose symbols at the next level using the right juxtapositions of symbols at the previous level, we need a systematic way of dealing with the combinatorial explosion. For example, if we use 1000 low level features obtained from BoW and create higher level symbols from just 2×2 patches of SIFT visual words, the potential combinatorial space of discrete symbols at next level is $\mathrm{O}(10^{12})$, clearly too prohibitive to just do traditional 2×2-gram histogram counting. If this were real-valued data, we could use any clustering technique but since this is symbolic data in a large vocabulary we need to use a non traditional clustering technique.

In this section we present a novel Naive Bayes clustering algorithm to cluster multi-variate discrete data in general and discrete image patches in particular. Note that in our experiments, we start with SIFT-BoW visual words; a discrete image patch is thus a patch of such visual words. To define a clustering algorithm, we need to define a "cost function", a "cluster representation" and of "update rules" to learn the cluster centers and cluster associations with data. First some notation: Let $\mathbf{X} = \{\mathbf{x}^n = (x_1^n \dots x_D^n)\}_{n=1}^N$ be the set of N data points. Each feature $X_d \in \mathbf{V}_d$ comes from a *discrete* feature vocabulary $\mathbf{V}_d = \left\{v_1^d \dots v_{M_d}^d\right\}$ of size $M_d = |\mathbf{V}_d|$. In image domain, each 2-D discrete image patch of size $P \times P$ is treated as a one-dimensional vector of size $D = P^2$ and each symbol comes from the same vocabulary, (i.e. $\mathbf{V}_d = \mathbf{V}$ of dense SIFT clusters.)

3.1 Mixture of Multi-variate discrete Naive Bayes

Mixture models [23] are commonly used to partition the data into meaningful clusters. Our patches are in $P \times P$ discrete space. Typically a parametric mixture model is learnt by maximizing a (log) likelihood objective over the data:

$$J(\mathbf{\Theta}) = \log \prod_{n=1}^{N} P(\mathbf{x}^n) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} P(k) P\left(\mathbf{x}^n | k\right). \tag{1}$$

Depending on the nature of the data, the mixture density function $P(\mathbf{x}|k)$ takes different forms. For example when $\mathbf{x} \in R^D$ any real-valued multi-variate density function such as a full Gaussian can be used. In our case $\mathbf{x} \in \mathbf{V}^D$ and therefore in this paper, we propose to use the simplest multi-variate discrete density function, i.e. *Naive Bayes* (NB):

$$P(\mathbf{x}^n | k) = \prod_{d=1}^{D} P\left(x_d^n | k\right) \tag{2}$$

In NB clustering, therefore we learn a “mixture-of-Naive Bayes” parametric generative model (Eq. 1) over a multi-variate discrete data by conveniently assuming independence among the features (Eq. 2). In general there are two constraints that are also part of the objective. The priors must add up to one and the density functions over all possible values that each feature can take for any given mixture component must also add up to one, i.e.,

$$\sum_{k=1}^{K} P(k) = 1, \sum_{m=1}^{M_d} P\left(v_m^d | k\right) = 1, \forall d = 1 \dots D \tag{3}$$

A total of $K \times \left(1 + \sum_{d=1}^{D} M_d\right)$ parameters $\mathbf{\Theta} = \left\{P(k), \left\{P(v_m^d)\right\}_{m=1}^{M_d}\right\}_{k=1}^{K}$ are learnt using an EM-algorithm with the following update rules for the E-step (Eq. 4) and smoothed M-step (Eq. 5 and 6) from iteration $t-1$ to iteration t.

$$P_t(k|\mathbf{x}_n) = \frac{P_{t-1}(\mathbf{x}_n|k) P_{t-1}(k)}{\sum_{k'}^{K} P_{t-1}(\mathbf{x}_n|k') P_{t-1}(k')} \tag{4}$$

$$P_t(k) = \frac{\lambda + \sum_{n=1}^{N} P_t(k|\mathbf{x}_n)}{\lambda K + N} \tag{5}$$

$$P_t(v_m^d|k) = \frac{\lambda' + \sum_{n=1}^{N} \delta(x_{n,d} = v_m^d) P_t(k|\mathbf{x}_n)}{\lambda' M_d + N P_t(k)} \tag{6}$$

Equation 4 computes the posterior probability of assigning a data point to cluster k in the next iteration (t) given the parameters at the previous iteration ($t-1$). Equations 5 and 6 are the parameter updates based on the assignment of data points to the clusters in this iteration. δ is the *kronecker delta*. Here we employ basic laplacian smoothing that takes affect mostly if the number of points in a cluster is small compared to the vocabulary size.

3.2 Soft vs. Hard Clustering

The EM algorithm described above represents a soft clustering algorithm where each data point is assigned to all clusters using the posterior probabilities i.e., $P_t(k|\mathbf{x}_n)$ in each iteration. This increases the computational complexity of the other update rules by a factor of K. In traditional (hard) clustering in each iteration, a data point is assigned to the cluster with the highest posterior probability. The hard clustering version of the above soft clustering algorithm alternates between the assign cluster E-step (Eq. 7) and the cluster parameters M-step (Eq. 8 and 9):

$$\kappa_{t-1}(\mathbf{x}^n) = \arg\max_{k=1\dots K} \{P_{t-1}(\mathbf{x}_n|k)P_{t-1}(k)\} \tag{7}$$

$$P_t(k) = \frac{1}{N}\sum_{n=1}^{N} \delta\left(\kappa_{t-1}(\mathbf{x}^n) = k\right) \tag{8}$$

$$P_t\left(v_m^d|k\right) = \frac{\sum_{n=1}^{N} \delta\left(x_d^n = v_m^d\right)\delta\left(\kappa_{t-1}(\mathbf{x}^n) = k\right)}{\sum_{n=1}^{N} \delta\left(\kappa_{t-1}(\mathbf{x}^n) = k\right)} \tag{9}$$

Hard clustering is faster since parameter updates take K times less time per iteration. Combined with a smarter initialization strategy discussed below, we found this to be better than soft clustering in terms of convergence and quality.

3.3 Smart Initialization

Sensitivity to initialization is a well known problem with clustering. Bad random initializations typically result in slow convergence, poor clustering quality and require multiple runs with different random initializations to generate the right final clusters. This randomness and uncertainty in clustering initialization can be mitigated by a number of smart initialization strategies [24]. In this paper, we employ a *farthest first point* (FFP) initialization. The goal of this initialization is to pick the initial K clusters such that they "cover" the entire data space well by spreading themselves as far away from each other as possible. Representation score of a point is defined as the similarity of a data point with the nearest cluster. The similarity between two data points $\mathbf{x}^n$ and $\mathbf{x}^{n'}$, $sim(\mathbf{x}^n, \mathbf{x}^{n'}) = \sum_k \delta(x_d^n = x_d^{n'})$ The FFP algorithm works as follows:

1. Initialize:
   – First cluster randomly: $k \leftarrow 1, \mu_1 = \mathbf{x}^r$ where r =random($\{1 \dots N\}$)
   – *Representation scores*: $R\left(\mathbf{x}^n|\mu_1\right) = sim(\mathbf{x}^n, \mu_1),\ \forall n = 1 \dots N$
2. Sample least represented point as the next cluster. If there are more than one equally representative points, pick one randomly.:

$$\mu_{k+1} = \arg\min_{n=1\dots N} R\left(\mathbf{x}^n|\mu_1 \dots \mu_k\right) \tag{10}$$

3. Update the representation scores of all data points:

$$R\left(\mathbf{x}^n|\mu_1 \dots \mu_{k+1}\right) = \max\left\{R\left(\mathbf{x}^n|\mu_1 \dots \mu_k\right), sim\left(\mathbf{x}^n, \mu_{k+1}\right)\right\}, \forall n = 1 \dots N$$

4. $k \leftarrow k + 1$, repeate steps 2 through 4 while $k < K$

FFP based smart initialization gives significantly better clusters and faster convergence than traditional random initializations.

4 Learning hierarchical bag of words

Any form of vector quantization gives a symbolic representation to the keypoints. Kmeans as a vector quantization framework has the limitation that it can cluster real valued keypoints only, because it has no distance metric to compare symbols. This is the primary hurdle that has prevented the evolution of models that learn hierarchical bags of features. As described in section 3.2, the NB clustering algorithm is designed to cluster symbolic data and hence it can be used to quantize discrete symbolic vectors. With this useful tool, we are prepared to exploit the principles of deep learning to learn features in the BoW domain.

4.1 Approach

We start conventionally by employing K-means on features computed at local image patches to give us symbol representations for the low level keypoints. Given these representations, we compute BoW representations of the images: we refer to these as our first level image representations / features. However, we do not lose the symbolic image yet, for it has spatial context. Adhering to the conventional mode of feature extraction, we collect keypoints (vectors of symbols) from patches in a dense grid over the level 1 symbol image. We quantize these symbolic vectors using the naive bayes clustering approach to get another level of symbols and another symbol image in turn. The symbols at this level are aggregations of the symbols at the previous level that lie in the same local neighbourhood. We compute the BoW representation of these level 2 symbol images and call these the second level image representations. We have thus devised a hierarchical feature extraction scheme that is independent of the way we get the visual words at any level: this process can be repeated any number of times to get a desired level of image representation. Figure 1 describes our approach.

4.2 Maximum Pooling

Spatial pooling is an idea borrowed from the deep learning community that introduces compactness in the representation and imparts invariance to distortions by reducing the spatial resolution. Conventionally, spatial pooling is done over a grid of cells where the keypoints within each cell are summarized by a single keypoint. For a cell c spanning $P \times P$ symbolic keypoints ($\mathbf{x}^n, n = 1, \ldots, P^2$), we define the cell representative α_c to be the symbol with the maximum posterior probability as given by

$$\alpha_c = \arg\max_n \{P(\mathbf{x}_n|\kappa(x^n))P(\kappa(x^n))\} \tag{11}$$

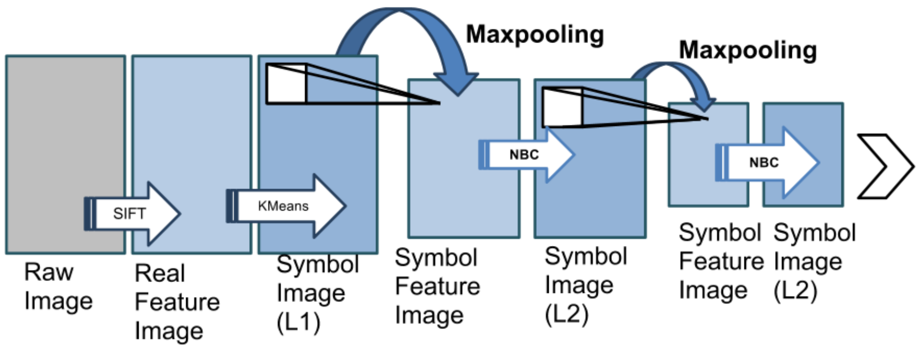


Fig. 1: Block diagram of our approach. SIFT features are computed on the raw image patches and quantized using K-means to get the first level symbol image. Henceforth, keypoints at any level of the hierarchy are collected from patches in a dense grid over the symbol image at the previous level. These keypoints are clustered using NB clustering and quantized to get the the symbol image at the current level. This process can be repeated any number of times. BoW representations can be computed using the symbol image at any level of the hierarchy and used for classification.

where $\kappa(x^n)$ is the cluster representative of x^n. In our experiments, we follow the usual convolutional network maxpooling protocol which uses non-overlapping patches of size 2×2 pixels.

5 Experiments, Results and Discussions

In this section, we use the NB clustering to learn hierarchical feature representations and use the learnt representations for the task of image classification. We first demonstrate our approach on a simple two class classification problem, and later show comprehensive results on two popular object classification datasets, namely Caltech 101 and Pascal VOC 2007. In this process, we gain insights into the learning by studying the effect of the parameters like the patch size (p), the size of the symbol space at each level (K) and the level of the hierarchy (l). We argue that the method learns semantically meaningful concepts by assessing the objective we are trying to achieve. We also demonstrate that hierarchical representations learnt through the NB clustering of visual words are better representations and outperform the traditional BoW method of image classification.
Experimental setup: In all the following experiments, we extract SIFT features over a dense grid using a scale of 12 and a shift of 6 pixels. Our baseline BoW representations are computed by clustering the SIFT vectors into 1000 visual words. For classification, we use a χ^2 homogeneous kernel map [3] on the BoW histograms and use a linear pegasos SVM [25].

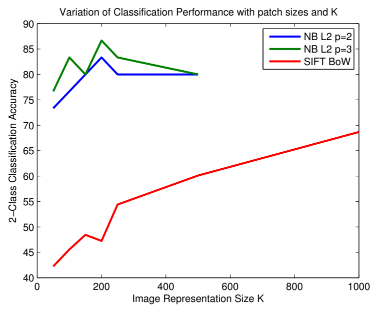
5.1 Two Class Classification: Okapi vs Llama

For the first set of experiments, the two classes we work with are llama and okapi (Figure 2a) which are part of the Caltech 101 dataset. We sample 15 images randomly for training and testing each from both these classes. This gives us training and testing sets of sizes 30 images each. These classes are hard to differentiate because the two animals look structurally similar and are found against similar looking backgrounds.

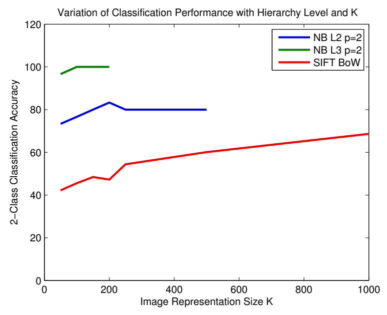
For these experiments, we use our baseline BoW representations to compute Level 2 features using patch sizes $p_2 = 2, 3$ with shifts 1 and 2 respectively and vocabulary size $K_2 = 50, 100, 150, 200, 250, 500$ (from this point on, we denote the patch size and the size of the symbol space at the n^{th} level by p_n and K_n respectively). BoW on the level 2 features given by each combination of p_2 and K_2 gives us a different level 2 representation. We compute Level 3 features using the level 2 representation given by $p_2 = 2, K_2 = 200$. For level 3, we use $p_3 = 2$ with shift of 1 and $K_3 = 50, 100, 200$. We use classification accuracy as the performance metric in our evaluations in these experiments.

Figures 2b and 2c compare the classification performance of hierarchical representations with the baseline BoW. In 2b we fix the representation level ($l = 2$) and vary K_2 and p_2; in 2c we fix p_2 and vary the representation level ($l = 2, 3$) and K_2 and K_3. In 2b, level 2 features significantly outperform level 1 features. Also, a patch size of 3 works gives better accuracy. At the representation size of 200, the performance gap between level 1 and level 2 features is 30%. In 2c The plots demonstrate the improvement in classification accuracies as we build higher representations. For level 3, we hit the 100% accuracy bound at $K_3 = 100$ while for level 1, accuracy is merely 68% for $K_1 = 1000$. Hence we achieve 32% higher accuracies using $1/10^{th}$ representation size.

(a)



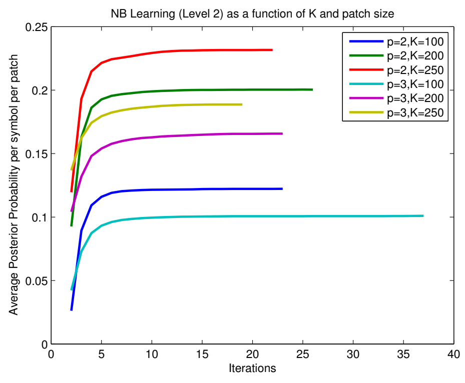
(b)



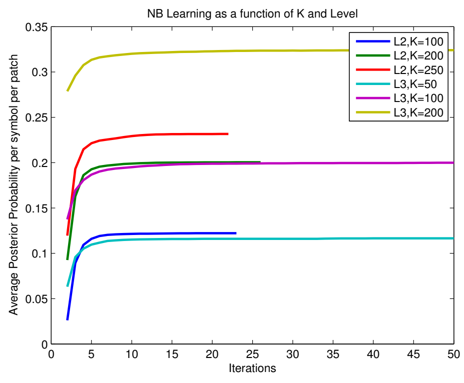
(c)

Fig. 2: Two-Class (Llama vs Okapi) Classification. (a) Llama (top) and Okapi (bottom) (b) Variation of accuracy with level 2 patch size and size of symbol space. (c) Classification accuracy based on Level 2,3 features; We chose p=2.

Figures 3a and 3b investigate what goes on at the core of the NB clustering algorithm during the vocabulary building procedure. We plot the average posterior probability per symbol per symbolic patch over the epochs of the training procedure. This is the average probability of a symbol to assume a particular position in a patch. This in turn determines the probability of a patch being part of a cluster of patches. In (c), we fix the level of representation ($l = 2$) while varying p_2 and K_2; in (d) we fix patch size and vary the representation level ($l = 2, 3$) and the number of clusters K_2 and K_3. It can be observed that bigger patches have lower average probabilities per symbol. This can be attributed to the fact that clustering a vector of 9 symbols is tougher than clustering a vector of 4 symbols because bigger patches are more complex in the number of ways the symbols are aligned in a patch. Another observation is that this probability increases as we increase the number of clusters. This can be explained by stating that increasing the number of clusters is allowing the arrangements of symbols in a patch more states to be in. Thus, each patch is more likely to find a state that it is most similar to. Finally, we comment on the increase in these probabilities across levels of the hierarchy. Figure 3b shows that the probabilities are higher for level 3 (for a fixed K). This shows that patches at this level are more likely (than patches at level 2) to find states / configuration of symbols that describe them. This has a direct bearing on the purity of the representations in terms of what they mean semantically.



(a)



(b)

Fig. 3: (a) Plot of mean posterior probabilities per symbol per patch over epochs. Effect of the patch size (p) and size of symbol space (K) can be seen here. (b) NB Learning for different sizes of symbol space (K) across hierarchical levels 2 and 3. Higher probabilities for Level 3 show that the method is learning semantically meaningful concepts.

Table 1: Classification accuracies on Caltech 101 for combinations of Spatial Pyramid and NB hierarchical features (and the baseline BoW). Table 2: Classification Accuracies on Caltech. BoW represents our baseline BoW results, BoW* represents the results quoted in [8]. Note that [8] uses pyramid kernel (and we use χ^2) and different scale, shift for SIFT computation. NBC represents our best results corresponding to L2, $K_2 = 250$ with level 2 SPM.

Table 1: Caltech 101- NB + SP

SPM	BoW	L2(250)	L3(200)
L 0	$43.4 \pm 1.2\%$	$60.4 \pm 0.7\%$	$61.3 \pm 1.4\%$
L 1	$59.0 \pm 0.8\%$	$68.2 \pm 0.8\%$	$66.6 \pm 1.6\%$
L 2	$68.3 \pm 1.3\%$	$72.4 \pm 0.6\%$	$69.8 \pm 0.9\%$
L 3	$67.6 \pm 0.7\%$	$67.8 \pm 1.1\%$	$66.3 \pm 1.4\%$

Table 2: Classification on Caltech

Method	Accuracy
BoW* [8]	$64.6 \pm 0.8\%$
CDBN [11]	$65.4 \pm 0.5\%$
BoW	$68.3 \pm 1.3\%$
NBC	$72.4 \pm 1.8\%$

5.2 Caltech 101

Caltech 101 contains a total of 9146 images, split among 101 distinct object categories. In these experiments, we sampled 30 random images for training from each of the 101 categories, getting a total of 3030 training images; the rest of the images were treated as testing images; however, as in [8], we limited the number of testing images per category to 50. These experiments were repeated 5 times with random subsampling and the mean classification accuracies over the five experiments are reported. To compute the BoW codebook, we sampled 5 training images from each category (505 images in all). We trained a one-vs-rest SVM for each class and the test image was assigned the label of the classifier with the highest score and report the accuracy of the classification.

For level 2, we use patches of sizes 2×2 and 3×3 with shifts of 1 and 2 pixels respectively. To compute the level 2 vocabulary, we sample 5 images randomly from each class and further sample each of these images to collect 25% of the total keypoints per image; we use $K_2 = 100, 250$. For the third level, the patch size is 2×2 with a shift of 1 and $K_3 = 100, 200$, vocabulary is computed using 5 random training images per class and using 25% of the keypoints per image. The classification pipeline remains the same as in the baseline case.

The classification accuracies for these procedures can be seen in Tables 1 and 2. It can be seen that hierarchical features learnt using the NB clustering approach significantly outperform the baseline BoW representation. Table 1 compares the classification performance of the baseline BoW representation with features at levels 2 and 3 and also shows further improvement in classification performance by using spatial pyramids on top of the image representations derived through the various methods. Hence, the representative power of hierarchical features can be further enhanced by using spatial pyramids. Note that in these experiments, we use only the spatial pyramid representation and not the pyramid matching kernel. As mentioned earlier, we use a homogeneous χ^2 kernel map; In Table 2, there are two rows devoted to BoW. BoW represents our baseline experiments (with χ^2 kernel), BoW* reports the accuraries quoted in [8] (Note that [8] uses SIFT features computed at a scale of 16 and shift of 8, and a pyramid matching kernel).

Kindly note that while the spatial pyramid representation size increases many folds per level, our hierarchical representations typically become more compact. We achieve better classification performance despite this fact.

5.3 Pascal VOC 2007

Pascal VOC 2007 data set has a total of 9955 images, split into 5011 training and 4944 testing images, distributed across 20 object categories. We use the entire dataset for our experiments. In these experiments, the BoW codebook was computed by clustering the keypoints collected from 10 random training images from each category (200 images in all). The classification scheme here is different from our experiments on Caltech. Pascal dataset allows multiple object categories in the same image, hence computing classification accuracies by assigning the class label of the classifier that returns the highest score to the test image is not fair assessment. In this set of experiments, we train a classifier for each class and compute Average Precision (AP) over the ranked list of test images. We finally report the mean AP over all the classes. Note that for our final image representation, we use a 2^{nd} level spatial pyramid [8]. As mentioned in section 1, deep learning methods have not enjoyed much success on this challenging dataset. Unlike Caltech 101, where the images are aligned and centered, Pascal has significant variation in the scale, position and orientation of the object in the image; it also allows multiple objects in the image. Note that CDBN [11] results are not available on Pascal 2007.

For level 2 features, we experiment with patch sizes of 2×2, 3×3 with shifts of 1 and 2 respectively. To compute the level 2 vocabulary, we sample 10 images randomly from each class and further sample each of these images to collect 25% of the total keypoints per image; we use $K_2 = 100, 250$. For level 3, we use patches of size 2×2 with a shift of 1 and $K_2 = 100, 200$. We use the same classification pipeline to classify the features at level 2 and 3. Table 3 displays the classification results for the mentioned methods. Here again, we significantly outperform the baseline results. While the baseline representation achieves a mean AP of 52.8% using a representation size of 1000, our L3 representation achieves 57% at a smaller representation size of 200. Hence, our representation is both richer and more compact.

Table 3: Classification Results on the Pascal VOC 2007 dataset. The table shows mean classification APs over 20 classes.

Method	SIFT BoW	L2	L2	L2	L3	L3	L3
p	-	3	3	2	2	2	2
K	1000	100	250	100	250	100	200
AP	52.84	54.90	55.86	55.64	56.20	56.48	57.04

5.4 Discussion

Our results on two classes explore the semantic meaning of the learnt hierarchical representations. Empirical comparisions with spatial pyramids reveal that we can achieve better classification accuracies with a much smaller representation size. Both these methods endow BoW with means to use spatial context. However, while spatial pyramids intend to discover the same low level artifacts in different regions of the image, we learn aggregates of such low level artifacts in the hope that these artifacts are part of a larger context and repeating this process of aggregation over and over will eventually lead us to learning the objects we are trying to classify.

The complexity of our representation is dictated by the number of levels in the hierarchy, the size of the patches and the size of the symbol space at each level. Deciding the optimal complexity requires all these parameters to be taken into account. This is similar to determining the number of clusters in clustering or any other model complexity determination problem. We believe this is still an open issue that may be addressed by empirical parameter sweeps or regularization theory as research in this field continues. In this paper, we experiment with 2-3 levels of hierarchy.

By empirically outperforming both SPM and CDBN representations on the two datasets, we demonstrate that our representations are both richer and more *compact*. For example, in Table 3, our L3 representation of size 200 outperforms the baseline representation of size 1000 significantly. The performance of our approach can be further improved by allowing a larger representation size, i.e. by using multiple scales of SIFT features as in [21]. In these experiments, our primary focus was to demonstrate the superiority of our representation over traditional BoW.

6 Conclusions

In this paper we devised a clustering framework for symbolic data points which can be used to learn hierarchical features starting with discrete data (such as BoW symbols.) Our method attempts to bridge the gap between two directions of research by developing a framework that learns from both approaches. We produce experimental evidence to argue that our hierarchical representations are semantically meaningful. We back this claim by outperforming the traditional BoW and deep learning representations on popular image classification datasets. It is quite possible that there are better distance functions between discrete features that may improve the learning procedure and the representative power of such a framework.

References

1. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV. (2003)

2. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: ECCV. (2004)
3. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. In: CVPR. (2010)
4. Perronnin, F., Snchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: IN: ECCV. (2010)
5. van Gemert, J.C., Geusebroek, J.M., Veenman, C.J., Smeulders, A.W.M.: Kernel codebooks for scene categorization. In: ECCV 2008, PART III. LNCS, Springer (2008) 696–709
6. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR. (2010)
7. Zhou, X., Yu, K., Zhang, T., Huang, T.S.: Image classification using super-vector coding of local image descriptors. In: ECCV. (2010)
8. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. (2006) 2169–2178
9. Hinton, G.E., Osindero, S., whye Teh, Y.: A fast learning algorithm for deep belief nets. In: Neural Computation. (2006)
10. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. (1998) 2278–2324
11. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: ICML. (2009)
12. Riesenhuber, M., Poggio, T., Studies, E.: Hierarchical models of object recognition in cortex (1999)
13. Fergus, R., Yu, K., Ranzato, M.A., Lee, H., Salakhutdinov, R., Taylor, G.: Tutorial on deep learning methods for vision. (In: CVPR 2012 Tutorial, http://cs.nyu.edu/˜fergus/tutorials/deep_learning_cvpr12/)
14. Agarwal, A., Triggs, B.: Multilevel image coding with hyperfeatures. In: International Journal of Computer Vision. (2008)
15. Quack, T., Ferrari, V., Leibe, B., Gool, L.V.: Efficient mining of frequent and distinctive feature configurations (2007)
16. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In: WGMBV. (2004)
17. Lazebnik, S., Schmid, C., Ponce, J.: A maximum entropy framework for part-based texture and object recognition. In: ICCV. (2005)
18. Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W.: Discovering objects and their location in images. In: ICCV. (2005)
19. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: ICCV. (2005)
20. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR. (2009)
21. Ken Chatfield, Victor Lempitsky, A.V., Zisserman, A.: The devil is in the details: an evaluation of recent feature encoding methods. In: BMVC. (2011)
22. lan Boreau, Y., Bach, F., Lecun, Y., Ponce, J.: Learning mid-level features for recognition. (2010)
23. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). (2007)
24. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: SODA. (2007)
25. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms (2008)