Multilevel Image Coding with Hyperfeatures

Ankur Agarwal, William Triggs

Multilevel Image Coding with Hyperfeatures

Ankur Agarwal and Bill Triggs
ankagar@microsoft.com, bill.triggs@inrialpes.fr

Abstract

Histograms of local appearance descriptors are a popular representation for visual recognition. They are highly discriminant with good resistance to local occlusions and to geometric and photometric variations, but they are not able to exploit spatial co-occurrence statistics over scales larger than the local input patches. We present a multilevel visual representation that remedies this. The starting point is the notion that to detect object parts in images, in practice it often suffices to detect co-occurrences of more local object fragments. This can be formalized by coding image patches against a codebook of known fragments or a more general statistical model and locally histogramming the resulting labels to capture their co-occurrence statistics. Local patch descriptors are converted into somewhat less local histograms over label occurrences. The histograms are themselves local descriptor vectors so the process can be iterated to code ever larger assemblies of object parts and increasingly abstract or 'semantic' image properties. We call these higher-level descriptors "hyperfeatures". We formulate the hyperfeature model and study its performance under several different image coding methods including k-means based Vector Quantization, Gaussian Mixtures, and combinations of these with Latent Dirichlet Allocation. We find that the resulting high-level features provide improved performance in several object image and texture image classification tasks.

Keywords: image coding, hierarchical representations, visual recognition, image classification

1 Introduction

One of the most popular representations for visual recognition is local coding using invariant local descriptors [39, 38, 2, 29, 10, 25, 26, 9, 35, 20, 13]. The image is treated as a loose collection of quasi-independent local patches, each patch is represented by a vector of robust visual descriptors, and a statistical summarization or aggregation process is used to capture the statistics of the resulting set of descriptor vectors. There are several variants for each component. Patches can be selected at one or at many scales, and either densely, at random, or sparsely according to local informativeness criteria [17, 21]. There are many types of local descriptors and they can incorporate various degrees of resistance to common perturbations such as viewpoint changes, geometric deformations, and photometric transformations [42, 29, 38, 31, 32]. Results can be aggregated in a variety of ways, either over local regions to make higher-level local descriptors, or globally to make whole-image descriptors.

The simplest example is the 'texton' or 'bag-of-features' approach. This was initially developed for texture analysis (*e.g.* [30, 28]), but it turns out to give surprisingly good performance in many image classification and object recognition tasks [45, 10, 9, 35, 20, 13]. Local image patches or their feature vectors are coded using vector quantization against a fixed codebook, and the votes for each codebook centre are tallied to produce a histogram characterizing the distribution of patches over the image or local region. Codebooks are typically constructed by running clustering algorithms such as k-means over large sets of training patches. Soft voting into several nearby centres can be used to reduce aliasing effects. More generally, EM can be used to learn a mixture distribution or a deeper latent variable model in descriptor space, coding each patch by its vector of posterior mixture-component membership probabilities or latent variable values.

1.1 Hyperfeatures

The main limitation of local coding approaches is that they capture only the first order statistics of the set of patches (within-patch statistics and their aggregates such as means, histograms, *etc.*), thus ignoring the fact that inter-patch statistics such as co-occurrences or spatial relationships are important for many recognition tasks. To alleviate this, several authors have proposed methods for incorporating an additional level of representation that captures pairwise or neighbourhood co-occurrences of coded patches [36, 40, 41, 2, 25].

This paper takes the notion of an additional level of representation one step further, generalizing it to a generic method for creating multi-level hierarchical codings. The basic idea is that image content should be coded at several levels of abstraction, with the higher levels being spatially coarser but (hopefully) semantically more informative. Our approach is based on local histograms, *c.f.* [36, 41]. At each level, the image is divided into local regions with each region being characterized by a descriptor vector. The base level contains raw image descriptors. At higher levels, each vector is produced by coding (*e.g.* vector quantizing) and locally pooling the finer-grained descriptor vectors from the preceding level. For instance, suppose that the regions at a particular level consist of a regular grid of overlapping patches that uniformly cover the image. Given an input descriptor vector for each member of this grid, the descriptors are vector quantized and their resulting codes are used to build local histograms of code values over (say) 5×5 blocks of input patches. These histograms are evaluated at each point on a coarser grid, so the resulting upper level output is again a grid of descriptor vectors (local histograms). The same process can be repeated at higher levels, at each stage taking a local set of descriptor vectors from the preceding level and returning its coded local histogram vector. We call the resulting higher-level features **hyperfeatures**. The codebooks are learned in the usual way, from descriptor vectors of the corresponding level taken from a set of training images. For improved scale-invariance, the process runs at each layer of a conventional multi-scale image pyramid, so each level of the hyperfeature hierarchy contains a whole pyramid of descriptor vectors, not just a grid[1]. The hyperfeature construction process is illustrated in fig. 1.

Our main claim is that hyperfeatures provide a useful set of features for visual recognition. In particular, the use of vector quantization coding followed by local histogramming of membership votes provides an effective means of integrating higher order spatial relationships into texton style image representations. The resulting spatial model is somewhat 'loose' – it codes nearby co-occurrences, not precise geometry – but for this reason it is robust to spatial misalignments and deformations and to partial occlusions, and it fits well with the "spatially weak / strong in appearance" philosophy of texton representations. The basic intuition is that in practice, notwithstanding its geometric weakness, simple co-occurrence of characteristic part fragments often suffices to characterize the presence of the parts, so that as one moves up the hyperfeature hierarchy larger and larger assemblies of parts are coded until ultimately one codes the entire object. Hyperfeature stacks are naturally robust to occlusions and feature extraction failures owing to their loose agglomerative nature. Even when the top level object is not successfully represented, substantial parts of it are often captured by the lower levels of the hierarchy and the system can still cue recognition on these.

1.2 Previous Work

This paper is an extended version of the ECCV paper [1]. The additional material includes the on-line version of the basic hyperfeature learning algorithm (§3) and more extensive experimental results.

The hyperfeature representation has several precursors. Classical 'texton' or 'bag of feature' representations [30, 28] are global histograms over quantized image descriptors – 'level 0' of the hyperfeature hierarchy. A hierarchical feature-matching framework for simple second level features

[1] Terminology: 'layer' denotes a standard image pyramid layer, *i.e.* the same image at a coarser scale; 'level' denotes the number of folds of hyperfeature (quantize-and-histogram) local coding that have been applied, with each transformation producing a different, higher-level 'image' or 'pyramid'.

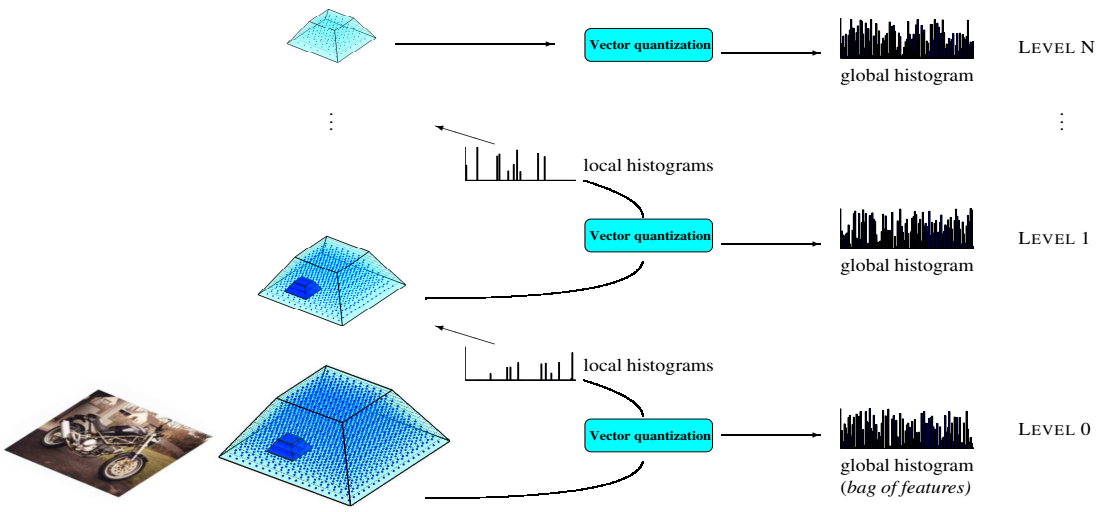


Figure 1: Constructing a hyperfeature stack. The 'level 0' (base feature) pyramid is constructed by calculating a local image descriptor vector for each patch in a multiscale pyramid of overlapping image patches. These vectors are vector quantized according to the level 0 codebook, and local histograms of codebook memberships are accumulated over local position-scale neighbourhoods (the smaller darkened regions within the image pyramids) to make the level 1 feature vectors. Other local accumulation schemes can be used, such as soft voting or more generally capturing local statistics using posterior membership probabilities of mixture components or similar latent models. Softer spatial windowing is also possible. The process simply repeats itself at higher levels. The level l to l+1 coding is also used to generate the level l output vectors – global histograms over the whole level-l pyramid. The collected output features are fed to a learning machine and used to classify the (local or global) image region.

is developed in [23], and histograms of quantized 'level 1' features are used to classify textures and recognize regularly textured objects in [36, 41].

Hyperfeature stacks also have analogies with multilevel neural models such as the neocognitron [16], Convolutional Neural Networks (CNN) [27] and HMAX [37, 43, 34]. These are all multilayer networks with alternating stages of linear filtering (banks of learned convolution filters for CNN's and of learned 'simple cells' for HMAX and the neocognitron) and nonlinear rectify-and-pool operations. The neocognitron activates a higher level cell if at least one associated lower level cell is active. In CNNs the rectified signals are pooled linearly, while in HMAX a max-like operation ('complex cell') is used so that only the dominant input is passed through to the next stage. The neocognitron and HMAX lay claims to biological plausibility whereas CNN is more of an engineering solution, but all three are convolution-based discriminatively-trained models. In contrast, although hyperfeatures are still bottom-up, they are essentially a descriptive statistics model not a discriminative one: training is completely unsupervised and there are no convolution weights to learn for hyperfeature extraction, although the object classes can still influence the coding indirectly via the choices of codebook. The basic nonlinearity is also different: for hyperfeatures, nonlinear descriptor coding by nearest neighbour lookup (or more generally by evaluating posterior probabilities of membership to latent feature classes) is followed by a comparatively linear accumulate-and-normalize process, while for the neural models linear convolution filtering is followed by simple but nonlinear rectification.

A number of other hierarchical feature representations have been proposed very recently. Some of these, *e.g.* [43, 34], follow the HMAX model of object recognition in the primal cortex while

others such as [11] are based on top-down methods of successively breaking top-level image fragments into smaller components to extract informative features. This does not allow higher levels to abstract from the features at lower levels. Another top-down approach presented in [24] is based on pyramid matching that repeatedly subdivides the image to compute histograms at various levels. Again, unlike the hyperfeatures that we propose in this paper, this does not feed lower level features into the computational process of higher ones. Hyperfeatures are literally "features of (local sets of) features", and this differentiates them from such methods. Another key distinction of hyperfeatures is that they are based on local co-occurence alone, not on numerical geometry (although quantitative spatial relationships are to some extent captured by coding sets of overlapping patches). Advantages of this are dicussed in section 3.

Note that the term 'hyperfeature' has also been used for several unrelated concepts including disjunctions of boolean features in document classification, and appearance descriptors augmented with feature position coordinates in visual recognition [14]. Our use of it was suggested by the local pooling performed by the 'hypercolumns' of the visual cortex.

2 Base Features and Image Coding

The hyperfeature framework can be used with a large class of underlying image coding schemes. This section discusses the schemes that were tested in this paper. For simplicity we describe them in the context of the base level (level 0), leaving the extension to higher levels to the next section.

2.1 Image Features

The 'level 0' input to the hyperfeature coder is a base set of local image descriptors. In our case these are computed on a dense grid – in fact a multiscale pyramid – of image patches. As patch descriptors we use SIFT-like gradient orientation histograms, computed in a manner similar to [29] but using a regularized normalization that remains finite for empty patches for better resistance to image noise in nearly empty patches. (SIFT was not originally designed to handle patches that may be empty). The normalization provides good resistance to photometric transformations, and the spatial quantization within SIFT provides a pixel or two of robustness to spatial shifts. The input to the hyperfeature coder in our experiments is thus a pyramid of 128-D SIFT descriptor vectors. Other descriptors could also be used, *e.g.* [33,3].

Hyperfeature models based on sparse (*e.g.* keypoint based [10,9,25,32]) feature sets would also be possible but they are not considered here, in part for simplicity and in part because recent work (*e.g.* [20]) suggests that dense representations will outperform sparse ones.

2.2 Vector Quantization and Gaussian Mixtures

Vector quantization is a simple and widely-used method of characterizing the content of image patches [28]. Each patch is coded by finding the most similar patch in a dictionary of reference patches and using the index of this patch as a label. Here we use nearest neighbour coding based on Euclidean distance between image descriptors, with a vocabulary learned from a training set using a clustering algorithm similar to the mean shift based on-line clusterer of [20]. We find that the results are relatively insensitive to the choice of method used to obtain this vocabulary. The histograms have a bin for each centre (dictionary element) that counts the number of patches assigned to the centre. In the implementation, a sparse vector representation is used for efficiency.

Abrupt quantization into discrete bins does cause some aliasing. This can be reduced by **soft vector quantization** – softly voting into the centers that lie close to the patch, *e.g.* with Gaussian weights. Taking this one step further, we can fit a probabilistic **mixture model** to the distribution of training patches in descriptor space, subsequently coding new patches by their vectors of posterior mixture-component membership probabilities. The full mixture model gives centres that are qualitatively very different from those obtained by clustering, as shown in figure

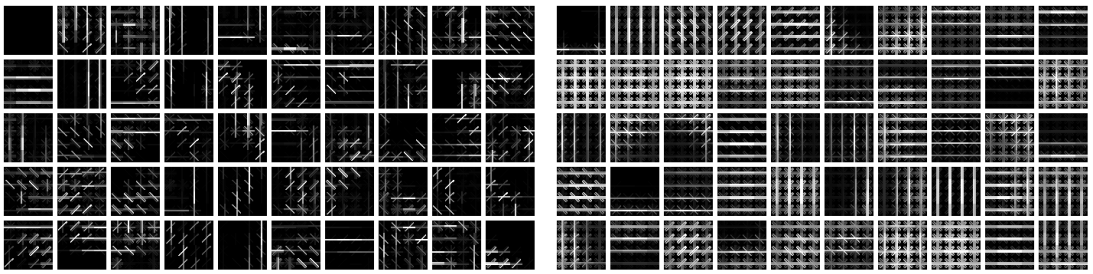


Figure 2: Codebook centers obtained for SIFT descriptors from a dataset of 684 images from 4 object categories (the VOC'05 dataset – see § 4). The intensity of each small line segment represents the weight of the corresponding orientation bin in the corresponding SIFT cell. emphLeft: Vector quantization cluster centers obtained using the mean-shift based clustering algorithm of [20]. *Right:* Gaussian mixture centres. The two codebooks clearly code information very differently. VQ picks relatively sparse 'natural features' while the GM tends to converge to denser, more averaged out features corresponding to structures such as vertical/horizontal lines, textures, *etc.*, relying to a large extent on variance differences to separate the different Gaussian components. The blank patch occurs very frequently in this dataset, especially in uncluttered background regions, and so it is almost always prominent among the centres.

2. In §4 we test hard vector quantization (VQ) and diagonal-covariance Gaussian mixtures (GM) fitted using Expectation-Maximization. The GM codings turn out to be more effective.

2.3 Latent Dirichlet Allocation

VQ and mixture models are flexible coding methods, but capturing fine distinctions often requires a great many centres. This brings the risk of fragmentation, with the patches of an object class becoming scattered over so many label classes that it is difficult to learn an effective recognition model for it. 'Bag of words' text representations face the same problem – there are many ways to express a given underlying 'meaning' in either words or images. To counter this, one can attempt to learn deeper latent structure models that capture the underlying semantic "topics" that generated the text or image elements. This improves learning because each topic label summarizes the 'meaning' of many different word labels.

The simplest latent model is Principal Components Analysis ('Latent Semantic Analysis' *i.e.* linear factor analysis), but in practice statistically-motivated nonlinear approaches such as Probabilistic Latent Semantic Analysis (pLSA) [18] perform better. There are many variants on pLSA, typically adding further layers of latent structure and/or sparsifying priors that ensure crisper distinctions [7, 8, 22, 6]. Here we use **Latent Dirichlet Allocation (LDA)** [4]. LDA models document words as samples from sparse mixtures of topics, where each topic is a mixture over word classes. More precisely: the gamut of possible topics is characterized by a learned matrix $\boldsymbol{\beta}$ of probabilities for each topic to generate each word class; for each new document a palette of topics (a sparse multinomial distribution) is generated using a Dirichlet prior; and for each word in the document a topic is sampled from the palette and a word class is sampled from the topic. Giving each document its own topic mixture allows more variety than sharing a single fixed mixture across all documents of a given class would, while still maintaining the underlying coherence of the topic-based structure. In practice the learned values of the Dirichlet parameter α are small, ensuring that the sampled topic palette is sparse for most documents. Pure LDA is an unsupervised generative topic model that does not use any discriminative training information such as knowledge of class labels. The process could be made more supervised by making the document topic priors class-specific, *e.g.* with distinct Dirichlet parameters for each topic. Although this might capture more discriminative information, here we use only standard unsupervised LDA.

Algorithm 1:

1. For all (i, x, y, s), $\mathcal{F}^{(0)}_{ixys} \leftarrow$ base feature at point (x, y), scale s in image i.
2. For $l = 0, \ldots, N$:
   - If learning, cluster $\{\mathcal{F}^{(l)}_{ixys} \,|\, \forall(i, x, y, s)\}$ to obtain a codebook of $d^{(l)}$ centres in this feature space.
   - For all i:
     - If $l < N$, for all (x, y, s) calculate $\mathcal{F}^{(l+1)}_{ixys}$ as a $d^{(l)}$ dimensional local histogram by accumulating votes from $\mathcal{F}^{(l)}_{ix'y's'}$ over neighbourhood $\mathcal{N}^{(l+1)}(x, y, s)$.
     - If global descriptors need to be output, code $\mathcal{F}^{(l)}_{i\ldots}$ as a $d^{(l)}$ dimensional histogram $\mathcal{H}^{(l)}_i$ by globally accumulating votes for the $d^{(l)}$ centers from all (x, y, s).
3. Return $\{\mathcal{H}^{(l)}_i \,|\, \forall i, l\}$.

Algorithm 2:

1. Initialization: run algorithm 1 using a small subset of the training images (*e.g.* 10)
2. Update codebook centres at all levels: for all i:
   - For $l = 0, \ldots, N$:
     - perform one iteration of k-means to update the $d^{(l)}$ centers using $\{\mathcal{F}^{(l)}_{ixys} \,|\, \forall(x, y, s)\}$.
     - If $l < N$, for all (x, y, s) calculate $\mathcal{F}^{(l+1)}_{ixys}$ as in algorithm 1.
3. If centers have not converged, go to step 2. Otherwise, for all i:
   - For $l = 0, \ldots, N$:
     - If $l < N$, for all (x, y, s) calculate $\mathcal{F}^{(l+1)}_{ixys}$ as a $d^{(l)}$ dimensional local histogram by accumulating votes from $\mathcal{F}^{(l)}_{ix'y's'}$ over neighbourhood $\mathcal{N}^{(l+1)}(x, y, s)$.
     - If global descriptors need to be output, code $\mathcal{F}^{(l)}_{i\ldots}$ as a $d^{(l)}$ dimensional histogram $\mathcal{H}^{(l)}_i$ by globally accumulating votes for the $d^{(l)}$ centers from all (x, y, s).
4. Return $\{\mathcal{H}^{(l)}_i \,|\, \forall i, l\}$.

Figure 3: Hyperfeature coding algorithms.

In our case, during both learning and use, the visual 'words' are represented by VQ or GM code vectors and LDA functions essentially as a locally adaptive nonlinear dimensionality reduction method, re-coding each word (VQ or GM vector) as a vector of posterior latent topic probabilities, conditioned on the local 'document' model (topic palette). The LDA 'documents' can be either complete images or the local regions used to construct (the current level of) the hyperfeature coding. Below we use local regions, which is slower but more discriminant. Henceforth, "coding" refers to either VQ or GM coding, optionally followed by LDA reduction.

3 Constructing Hyperfeatures

The hyperfeature construction process is illustrated in figure 1. At level 0, the image (more precisely the image pyramid) is divided into overlapping local neighbourhoods, with each neighbourhood

containing a number of image patches (possibly across multiple scale layers in the pyramid). The co-occurrence statistics within each local neighbourhood $\mathcal{N}$ are captured by vector quantizing or otherwise nonlinearly coding its patches and histogramming the results over the neighbourhood. This process converts local patch-level descriptor vectors (image features) to spatially coarser but higher-level neighbourhood-level descriptor vectors (local histograms). It works for any kind of descriptor vector. In particular, the output is again a vector of local image descriptors so the process can be repeated recursively over higher and higher order neighbourhoods to obtain a series of increasingly higher level but spatially coarser descriptor vectors.

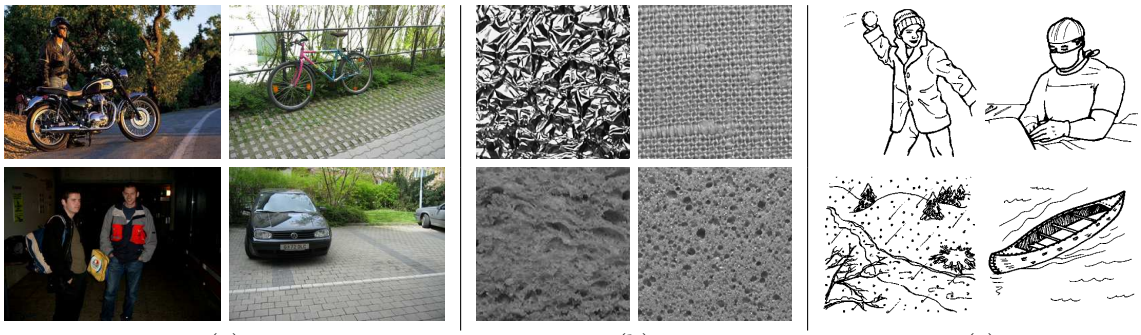
Let: $\mathcal{F}^{(l)}$ denote the level-l hyperfeature pyramid; $d^{(l)}$ denote the level-l feature, codebook or histogram dimension; (x, y, s) denote position-scale coordinates within a feature pyramid; and $\mathcal{F}^{(l)}_{ixys}$ denote the level-l descriptor vector at (x, y, s) in image i. During training, a codebook or coding model is learned from all features (all i, x, y, s) at level l. In use, the level-l codebook is used to code the level-l features in some image i, and these are pooled spatially over local neighbourhoods $\mathcal{N}^{(l+1)}(x, y, s)$ to make the hyperfeatures $\mathcal{F}^{(l+1)}_{ixys}$. In [1], we developed a batch-mode hyperfeature coding algorithm that learns the coding one level at a time. The algorithm is simple to implement but it becomes too memory-intensive for large datasets because for batch VQ, E-M or LDA, at each level l, the (densely sampled) level-l features from all images need to be stored in live memory during training. The batch algorithm for VQ coding on N levels is summarized as 'algorithm 1' in figure 3. In this paper we introduce an alternative algorithm that processes the images sequentially on-line, simultaneously learning codings for all levels – see algorithm 2 in figure 3. The two algorithms give very similar results (*e.g.* for test image classification error on the 4-class dataset below), but the second can handle arbitrarily large sets of training images because it only needs to store and process one image at a time. In principle the on-line algorithm is asymptotically slower than the batch one because it processes every layer in each iteration and no layer can fully converge until the previous ones have. However in practice on-line k-means has rapid initial convergence and we typically find that all of the layers have stabilized before the first pass through the data has been completed. In the experiments below, performing only step 1 of algorithm 2 already gives acceptable results, one pass of step 2 improves the classification performance by 1-2% on average, and further passes make little difference. So overall the run times of the two methods are very similar.

For vector quantization coding there is a single global clustering during learning, followed by local histogramming of class labels within each neighbourhood during use. For GM coding, a global mixture model is learned using EM, and in use the mixture component membership probability vectors of the patches from the neighbourhood are summed to get the code vector. In the on-line GM algorithm, step 2 involves updating the covariances along with the centers. This is done in the usual on-line EM manner by storing sufficient statistics (weighted sums of number of points, point coordinates and their squares of all points assigned to the cluster so far).

If LDA is used, each neighbourhood is treated as a separate 'document' with its own LDA context in the experiments below, *i.e.* the parameters $\alpha, \boldsymbol{\beta}$ are estimated once using all of the training images, then used as priors for the topic distributions of each neighbourhood independently.

In all of these schemes, the histogram dimension is the size of the codebook or the GM/LDA basis. The neighbourhoods are implemented as small trapezoids in scale space, as shown in figure 1. This shape maintains scale invariance and helps to minimize boundary losses, which cause the pyramids to shrink in size with increasing level. The size of the pooling region at each hyperfeature level (relative to its corresponding source layer in the original input pyramid) is a parameter. The region size needs to grow with the level. Otherwise, essentially the same information is re-encoded at each level, causing rapid saturation and suboptimal performance.

As tested here, hyperfeature coding is a purely bottom-up descriptive statistics approach, trained without any reference to class labels. It thus offers a relatively neutral and generic framework for image representation that is not explicitly specialized for discrimination between particular classes of image content. Below we will study the effectiveness of hyperfeatures for image classification and region labelling. This is done by training a separate discriminative classifier on labelled images after the generic hyperfeature coding has been learned. The classifiers take the multi-level



(a) (b) (c)

Figure 4: Some typical images from the datasets used to evaluate hyperfeature based image classification. (a) The VOC'05 object dataset contains 4 classes: motorbikes, bicycles, people and cars; (b) 4 of the 10 different textures in the KTH-TIPS dataset; and (c) the CRL-IPNP dataset includes drawings of people, objects and scenes.

hyperfeatures as input and output predicted class labels for each image (§4) or for each local image region (§5).

4 Experiments on Image Classification

To illustrate the discriminative capabilities of hyperfeatures, we present image classification experiments on three datasets: a 4 class object dataset based on the "Caltech 7" and "Graz" datasets that was used for the European network PASCAL's "Visual Object Classes Challenge 2005" (VOC'05) [12]; the 10 class KTH-TIPS texture dataset [15]; and the CRL-IPNP (Centre of Research in Language – International Picture Naming Project) dataset of line sketches used for picture naming in language research[2]. The VOC'05 dataset contains 684 training and 689 test images, which we scale to a maximum resolution of 320×240 pixels. The texture dataset contains 450 training and 360 test images over 10 texture classes, mostly 200×200 pixels. The CRL-IPNP dataset consists of 360 images of 300×300 pixels picturing various common objects and activities. Here we divide it into just two classes, images containing people, and others.

As base level visual features we use the underlying descriptor of Lowe's SIFT method – local histograms of oriented image gradients calculated over 4×4 blocks of 4×4 pixel cells [29]. Here this is tiled densely over the image with no orientation normalization, not applied sparsely at keypoints and rotated to the dominant local orientation [29], as orientation normalization actually reduces the classification performance on the objects dataset. The input pyramid had a scale range of 8:1 with a spacing of 1/3 octave. Patches were sampled regularly at 8 pixel intervals, giving a total of 2500–3000 descriptors per image. For the pooling neighbourhoods $\mathcal{N}$, we took volumes of 3×3×3 patches in (x, y, s) by default, increasing these in effective size by a factor of $2^{1/3}$ (one pyramid layer) at each hyperfeature level.

Classification Framework: With all of the schemes that we tested for hyperfeature construction, the features are learned in a completely unsupervised manner. To perform image classification using these features, we train soft linear one-against-rest Support Vector Machine classifiers [44] independently for each class over the global output histograms collected from the active hyperfeature levels. We compare the performance using different numbers of hyperfeature levels, in each case using the entire set of global histograms from the stated level and from all lower ones as features. The experiments here use SVM-light [19] with default settings.

[2]The VOC'05 object recognition database collection is available at `www.pascal-network.org/challenges/VOC`. The KTH-TIPS texture dataset is available at `www.nada.kth.se/cvap/databases/kth-tips`. The CRL-IPNP dataset is available at `http://crl.ucsd.edu/~aszekely/ipnp`.

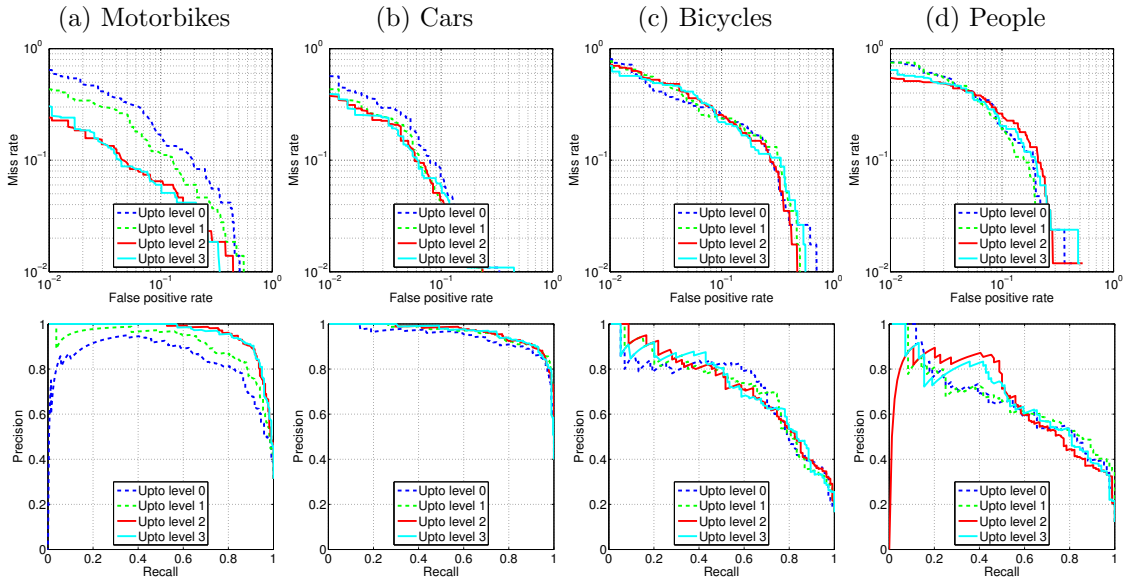


Figure 5: Detection Error Trade-off and Recall-Precision curves for the four classes of the VOC'05 dataset. Up to a certain level, including additional levels of hyperfeatures improves the classification performance. For the motorbike, car and bicycle classes the best performance is at level 3, while for the person class it is at level 1 (one level above the base features). The large gain on the motorbike (a 5× reduction in false positives at fixed miss rate) and car classes suggests that local co-occurrence structure is quite informative, and is captured well by hyperfeatures.

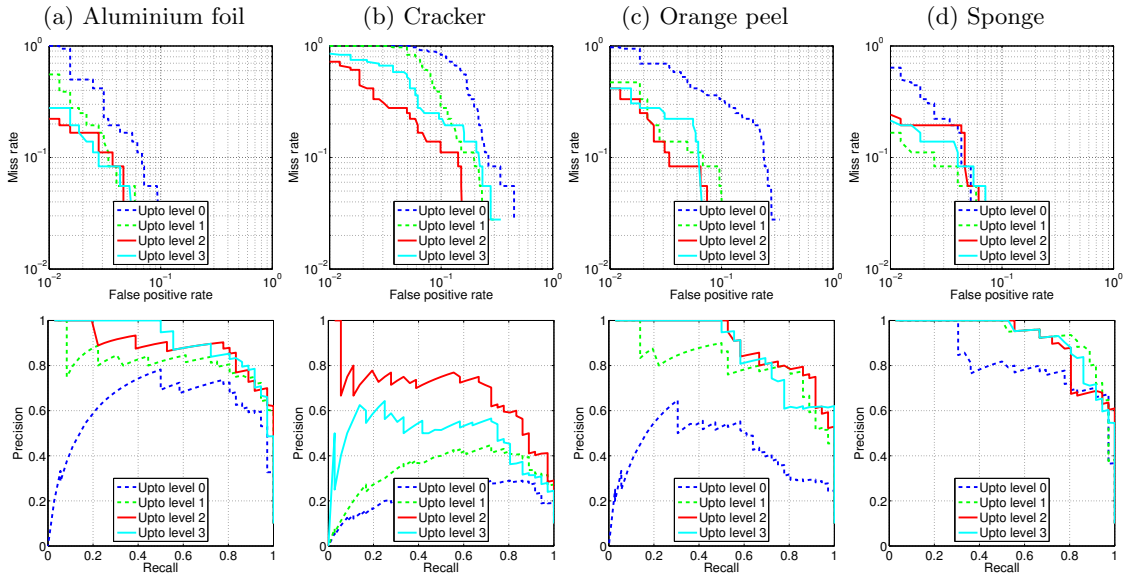


Figure 6: Detection Error Trade-off and Precision-Recall curves for 4 of the 10 classes of the KTH-TIPS dataset for which hyperfeatures play a significant role (see text). A mixture of 100 Gaussians is used at each level. Including hyperfeatures improves the classification performance for every texture that is poorly classified at level 0, without hurting that for well-classified textures. The aluminium and sponge classes are best classified by including 3 levels of hyperfeatures, and cracker and orange peel by using 2 levels.

	Al. foil	Bread	Corduroy	Cotton	Cracker	Linen	Orange peel	Sandpaper	Sponge	Styrofoam
VQ	97.2	88.1	100	86.1	94.4	77.8	94.4	83.3	91.7	88.9
GM	100	88.9	100	88.9	91.6	86.1	94.4	83.3	91.7	91.7

Figure 7: One-vs-rest classification performance (hit rate) at the equal error point for the 10 classes of the texture dataset, using hard vector quantization (VQ) and a diagonal Gaussian mixture model learned by EM (GM). Each class uses its optimal number of hyperfeature levels. GM performs best on average.

4.1 The Effect of Multiple Levels

Figure 5 presents Detection Error Tradeoff (DET)[3] and Precision-Recall curves showing the influence of hyperfeature levels on classification performance for the VOC'05 dataset. We used GM coding with a 200 center codebook at the base level and 100 center ones at higher levels. Including higher levels gives significant gains for 'cars' and especially 'motorbikes', but little improvement for 'bicycles' and 'people'. The results improve up to level 3 (*i.e.* using the hyperfeatures from all levels 0–3 for classification), except for 'people' where level 1 is best. Beyond this there is overfitting: subsequent levels introduce more noise than information. The differences in behaviour between classes can probably be attributed to their differing amounts of *structure*. The large appearance variations in the 'person' class leave little in the way of regular co-occurrence statistics for the hyperfeature coding to key on, whereas the more regular geometries of cars and motorbikes are captured well, as seen in figure 5(a) and (b). Different coding methods and codebook sizes give qualitatively similar evolutions with level, but the absolute numbers can be quite different (see below).

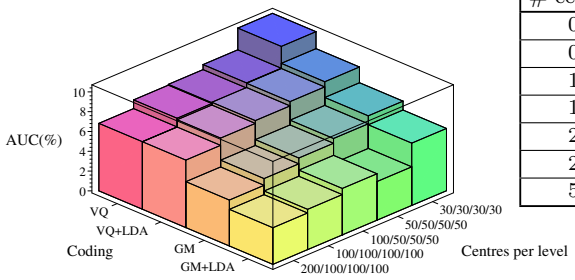
The results on the KTH-TIPS texture dataset lead to similar conclusions. For 4 of the 10 classes the level 0 performance is already near perfect and adding hyperfeatures makes little difference. For the remaining 6, there are gains (often substantial ones) up to hyperfeature level 3. Curves for 4 of these 6 classes are shown in figure 6. The equal-error-rate texture classification performance[4] for all 10 classes for VQ and GM coding is shown in figure 7. GM is better on average. Overall, its mean hit rate of 91.7% at equal error rate is slightly better than the 90.6% achieved by the bank of filters approach in [15] – a good result considering that these experiments used relatively few centres, widely spaced samples and only a linear SVM.

4.2 Coding methods and hyperfeatures

Figure 8 shows average miss rates (*i.e.* Area Under the DET Curve or 1− Area Under ROC Curve) on the VOC'05 dataset, for different coding methods and numbers of centers. The overall performance depends considerably on both the coding method used and the codebook size (number of clusters / mixture components / latent topics), with GM coding dominating VQ, the addition of LDA always improving the results, and performance increasing whenever the codebook at any level is expanded. On the negative side, learning large codebooks is computationally expensive, especially for GM and LDA. GM gives much smoother codings than VQ as there are fewer aliasing artifacts, and its partition of the descriptor space is also qualitatively very different – the Gaussians overlap heavily and inter-component differences are determined more by covariance differences than by centre differences. LDA seems to be able to capture canonical neighbourhood structures more

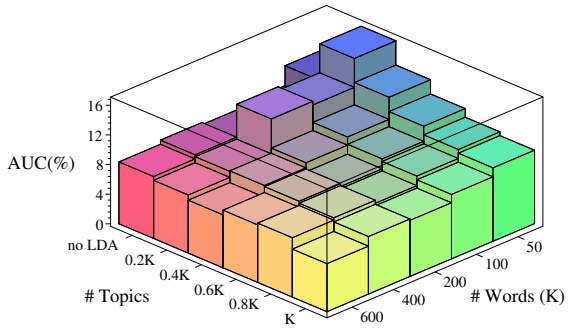
[3] DET curves plot miss rate (*i.e.* 1−true positive rate) *vs.* false positive rate on a log-log scale – the same information as a ROC curve in more visible form. Lower curves indicate better performance.

[4] At the base level of the texture dataset, we needed to make a manual correction to the SIFT VQ codebook to work around a weakness of codebook creation. Even at the finest level at which SIFT's are evaluated here, certain textures are homogeneous enough to cause all bins of the SIFT descriptor to fire about equally. This gives rise to a very heavily populated "uniform noise" centre in the middle of SIFT space. For some textures this centre receives nearly all of the votes, significantly weakening the base level coding and thus damaging the performance at all levels. The issue can be resolved by simply deleting the rogue centre (stop word removal). It does not occur either at higher levels or for GM coding.



# centers, levels 0-1-2-3	VQ	VQ+LDA	GM	GM+LDA
030-030-030-030	10.49	8.78	7.19	6.29
050-050-050-050	8.29	8.26	6.39	3.90
100-050-050-050	7.78	7.83	5.91	4.69
100-100-100-100	7.70	7.55	5.32	4.07
200-100-100-100	6.82	6.82	4.64	**3.70**
200-200-200-200	**6.55**	6.21	**4.71**	–
500-100-100-100	6.73	**5.76**	5.75	–

Figure 8: Average miss rates (area under DET curve – AUC) for different codebook sizes and coding methods on the VOC'05 object test set. Increasing the codebook size at any level almost always improves the performance. GM coding outperforms VQ coding even with significantly fewer centres, and adding LDA consistently improves the results. The LDA experiments shown here use the same number of topics as VQ/GM codebook centres, so they do not change the dimensionality of the code, but they do make it sparser.



# centers at levels 0-1-2-3	Train	Test
600-000-000-000	7.46	8.35
300-300-000-000	5.30	7.77
400-200-000-000	**4.06**	7.41
200-200-200-000	4.24	**6.57**
150-150-150-150	4.35	7.30

Figure 9: *Left:* The effect of LDA on average classification performance on test images – average miss rates for the VOC'05 objects testset. The performance improves systematically as both code centres (here VQ) and LDA topics are added. It continues to improve even when there are more LDA topics than codebook centres, presumably owing to improved sparsity. *Right:* For a fixed total number of centers (here VQ ones), performance improves if they are distributed relatively evenly across several levels – here 3, with the inclusion of a 4^{th} reducing the performance. *I.e.* adding higher level information is more useful than adding finer-grained low level information.

crisply than VQ or GM, presumably because it codes them by selecting a sparse palette of topics rather than an arbitrary vector of codes. If used to reduce dimensionality, LDA may also help simply by reducing noise or overfitting associated with large VQ or GM codebooks.

Given that performance always improves with codebook size, one could argue that rather than adding hyperfeature levels, it may be better to include additional base level features. To study this we fixed the total coding complexity at 600 centres and distributed the centres in different ways across levels. Figure 9 (right) shows that spreading centres relatively evenly across levels (here up to level 3) improves the results, confirming the importance of higher levels of abstraction.

4.3 Extent of Local Spatial Aggregation

As detailed in §3, hyperfeature construction involves aggregating descriptor statistics over local neighbourhood regions $\mathcal{N}$ at each level. It might be useful to automatically determine the optimal extent of these regions, allowing them to adapt in size depending on the spatial statistics of the descriptors at the current level, but here we assume fixed region sizes. Figure 10 shows the effect of using pooling neighbourhoods larger than the default size of 3×3×3 descriptors on the CRL-IPNP dataset with a 100-centre codebook at each level. The optimal performance is attained at a certain (modest and dataset dependent) neighbourhood size. The optimal number of hyperfeature

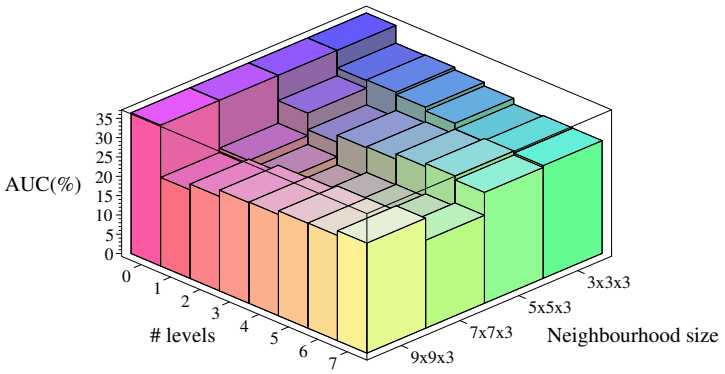


Figure 10: Performance on the CRL-IPNP dataset: average miss rates on the positive class for different pooling neighbourhood sizes and different numbers of hyperfeature levels. For a 3×3×3 neighbourhood (in x, y, s), 5 levels of hyperfeatures are best, but the best overall performance is achieved by 7×7×3 neighbourhoods with 3 levels of hyperfeatures.

levels also depends on this size: as the pooling regions grow, fewer hyperfeature levels are typically required. For the CRL-IPNP dataset, a spatial extent of 7×7×1 with 3 levels of hyperfeatures proves to be optimal.

5 Object Localization

One advantage of hyperfeatures is that they offer a controllable trade-off between locality and level of abstraction: higher level features accumulate information from larger image regions and thus have less locality but potentially more representational power. However the optimal pooling regions prove to have relatively modest sizes, so even quite high-level hyperfeatures can still be local enough to provide useful object-region level image labeling. Here we exploit this for the bottom-up localization of objects of interest. The image pyramid is tiled with regions and in each region we build a mini-pyramid containing the region's hyperfeatures (*i.e.* the hyperfeatures of all levels, positions and scales whose support lies entirely within the region). The resulting region-level hyperfeature histograms are then used to learn a local region-level classifier for each class of interest. Our goal here is simply to illustrate the representational power of hyperfeatures, not to build a complete object recognition system, so in the experiments below we will classify regions individually without any attempt to exploit spatial contiguity or top-down information.

The experiments shown here use the bounding boxes provided with the VOC'05 dataset as object masks for foreground labeling[5]. The foreground labels are used to train linear SVM classifiers over the region histograms, one for each class with all background and other-class regions being treated as negatives. Figure 11 shows some typical results obtained using these one-against-the-rest classifiers. Even though each patch is treated independently, the final labellings are coherent enough to allow the objects to be loosely localized in the images. The average accuracy of local region classification over all classes is 69%. This is significantly lower than the performance for classifying images as a whole, but still good enough to be useful as a bottom-up input to higher-level visual routines. Hyperfeatures again add discriminative power to the base level features, improving region level labelling accuracy by 4–5% on average. Figure 12 shows the key entries of the combined and the two-class confusion matrices, with negatives being further broken down into

[5]This labeling is not perfect. For many training objects, the bounding rectangles contain substantial areas of background, which are thus effectively labeled as foreground. Objects of one class also occur unlabeled in the backgrounds of other classes and, *e.g.*, instances of people sitting on motorbikes are labeled as 'motorbike' not 'person'. In the experiments, these imperfections lead to some visible 'leakage' of labels. We would expect a more consistent foreground labeling to reduce this significantly.

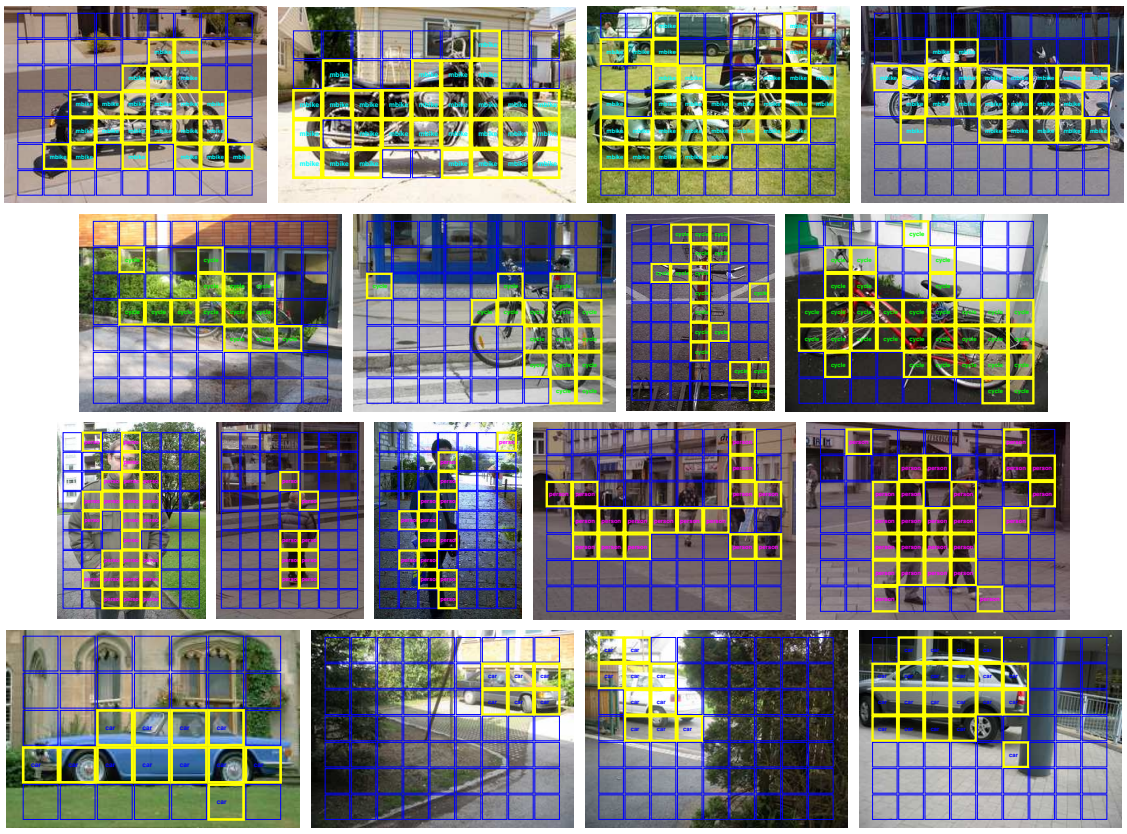


Figure 11: Object localization in the VOC'05 dataset by classifying local image regions using hyperfeatures. Each row shows examples of results using one of the four independent classifiers, each being trained to classify foreground regions of its own class against the combined set of all other regions – background regions and foregrounds from other classes. An image region is labeled as belonging to the object class if the corresponding SVM returns a positive score. We make no attempt to enforce spatial coherence here: each region is classified independently.

true \ estimated	motorbike	cycle	person	car	background
motorbike	**41.02**	17.58	10.03	18.02	13.34
cycle	20.17	**42.21**	14.66	6.51	16.45
person	9.81	13.67	**55.71**	6.43	14.39
car	18.32	4.56	6.19	**63.00**	7.93
background	7.48	13.66	15.99	19.09	**43.78**
true proportion	20.62	9.50	3.52	4.71	61.65

true \ est.	motorbike	cycle	person	car
motorbike	**69.34**	45.17	19.79	35.76
cycle	49.82	**63.56**	26.08	14.43
person	27.01	35.37	**65.84**	19.54
car	52.43	12.43	10.39	**77.30**
background	16.36	19.81	19.46	23.46
negative	**22.98**	**25.81**	**19.74**	**25.07**

Figure 12: Confusion matrices for region level labeling. Four two-class linear SVM region classifiers are trained independently, each treating regions from the background and from other classes as negatives. *Left:* A classical confusion matrix for the classifiers in winner-takes-all mode with patches that have negative scores for all object classifiers being labelled as background. The final row gives the population proportions, *i.e.* the score for a random classifier. *Right:* Each column gives entries from the pairwise confusion matrix of the corresponding classifier used alone (independently of the others), with the negative true-class scores (final row) broken down into scores on each other class and on the background. (NB: in this mode, the assigned class labels are not mutually exclusive).

true background patches and patches from the three remaining classes.

6 Conclusions and Future Work

We have introduced *hyperfeatures*, a new multilevel nonlinear image coding mechanism that generalizes – or more precisely, iterates – the quantize-and-vote process used to create local histograms in texton / bag-of-feature style approaches. Unlike previous multilevel representations such as convolutional neural networks and HMAX, hyperfeatures are optimized for capturing and coding local appearance patches and their co-occurrence statistics. The framework can employ any local coding method at the individual levels. It allows some of the spatial structure of the image to be captured and exploited without assuming rigidity or any strong geometric model. Our experiments show that the introduction of one or more levels of hyperfeatures improves the performance in many classification tasks, especially for object classes that have distinctive geometric or co-occurrence structures.

The hyperfeature idea is applicable to a wide range of problems involving part-based representations. In this paper the hyperfeatures were trained bottom-up by unsupervised clustering, but more discriminative training methods should be a fruitful area for future investigation. More focused codebooks could probably be learned by including priors that encourage object-specific coding and by incorporating image class labels into the learning of the latent topics. More general LDA like methods that use local context while training could be investigated. One way to do this is to formally introduce a "region" (or "subdocument") level in the word–topic–document hierarchy. Such models should allow us to model contextual information at several different levels of support, which may be useful for object detection.

We have seen that hyperfeatures are very discriminative even on local image regions. This suggests that hyperfeature based local region classifications could be used to bootstrap more classical sliding-window object detectors. Also, combining hyperfeature based representations of local image regions with Markov or conditional random fields would allow for improved object localization in images.

Acknowledgments

We would like to thank the European projects LAVA and PASCAL for financial support, and Diane Larlus, Frederic Jurie, Gyuri Dorko and Navneet Dalal for comments and code. We are also grateful to Andrew Zisserman and Jitendra Malik for providing feedback on this work. Our experiments make use of code derived from C. Bouman's *Cluster* [5] for fitting Gaussian mixtures, D. Blei's implementation of LDA [4], and T. Joachim's SVM-Light [19].

References

[1] A. Agarwal and B. Triggs. Hyperfeatures – Multilevel Local Coding for Visual Recognition. In *European Conf. Computer Vision*, pages 30–43, 2006.

[2] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *PAMI*, 26(11):1475–1490, November 2004.

[3] A. Berg and J. Malik. Geometric Blur for Template Matching. In *Int. Conf. Computer Vision & Pattern Recognition*, 2001.

[4] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] C. A. Bouman. Cluster: An unsupervised algorithm for modeling Gaussian mixtures. Available from http://www.ece.purdue.edu/~bouman, April 1997.

[6] W. Buntine and A. Jakaulin. Discrete principal component analysis. Technical report, HIIT, 2005.

[7] W. Buntine and S. Perttu. Is multinomial pca multi-faceted clustering or dimensionality reduction? *AI and Statistics*, 2003.

[8] J. Canny. GaP: A factor model for discrete data. In *ACM Conference on Information Retrieval (SIGIR)*, Sheffield, U.K., 2004.

[9] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *European Conf. Computer Vision*, 2004.

[10] G. Dorko and C. Schmid. Object class recognition using discriminative local features. Technical report, INRIA Rhône Alpes, 2005.

[11] M. Everingham *et.al.* The 2005 PASCAL visual object classes challenge. In F. d'Alche Buc, I. Dagan, and J. Quinonero, editors, *Proceedings of the first PASCAL Challenges Workshop*. Springer LNAI, 2006.

[12] B. Epshtein and S. Ullman. Feature Hierarchies for Object Classification. In *Int. Conf. Computer Vision*, 2005.

[13] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Int. Conf. Computer Vision & Pattern Recognition*, 2005.

[14] A. Ferencz, E. Learned-Miller, and J. Malik. Learning Hyper-Features for Visual Identification. In *Neural Information Processing Systems*, 2004.

[15] M. Fritz, E. Hayman, B. Caputo, and J.-O. Eklundh. On the Significance of Real-World Conditions for Material Classification. In *European Conf. Computer Vision*, 2004.

[16] K. Fukushima. Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics*, 36(4):193–202, 1980.

[17] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[18] T. Hofmann. Probabilistic Latent Semantic Analysis. In *Proc. of Uncertainty in Artificial Intelligence*, Stockholm, 1999.

[19] T. Joachims. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[20] F. Jurie and B. Triggs. Creating Efficient Codebooks for Visual Recognition. In *Int. Conf. Computer Vision*, 2005.

[21] T. Kadir and M. Brady. Saliency, Scale and Image Description. *Int. J. Computer Vision*, 45(2):83–105, 2001.

[22] M. Keller and S. Bengio. Theme-Topic Mixture Model for Document Representation. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining*, 2004.

[23] G. Lang and P. Seitz. Robust Classification of Arbitrary Object Classes Based on Hierarchical Spatial Feature-Matching. *Machine Vision and Applications*, 10(3):123–135, 1997.

[24] S. Lazebnik, C. Schimd, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Int. Conf. Computer Vision & Pattern Recognition*, 2006.

[25] S. Lazebnik, C. Schmid, and J. Ponce. Affine-Invariant Local Descriptors and Neighborhood Statistics for Texture Recognition. In *Int. Conf. Computer Vision*, 2003.

[26] S. Lazebnik, C. Schmid, and J. Ponce. Semi-local Affine Parts for Object Recognition. In *British Machine Vision Conference*, volume volume 2, pages 779–788, 2004.

[27] Y. LeCun, F.-J. Huang, and L. Bottou. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *CVPR*, 2004.

[28] T. Leung and J. Malik. Recognizing Surfaces Using Three-Dimensional Textons. In *Int. Conf. Computer Vision*, 1999.

[29] D. Lowe. Distinctive Image Features from Scale-invariant Keypoints. *Int. J. Computer Vision*, 60, 2:91–110, 2004.

[30] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *J. Optical Society of America*, A 7(5):923–932, May 1990.

[31] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 27(10), 2005.

[32] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65(1/2), 2005.

[33] G. Mori and J. Malik. Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. In *Int. Conf. Computer Vision & Pattern Recognition*, 2003.

[34] J. Mutch and D. Lowe. Multiclass object recognition with sparse, localized features. In *Int. Conf. Computer Vision & Pattern Recognition*, pages I 11–18, 2006.

[35] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. In *European Conf. Computer Vision*, 2004.

[36] J. Puzicha, T. Hofmann, and J. Buhmann. Histogram Clustering for Unsupervised Segmentation and Image Retrieval. *Pattern Recognition Letters*, 20:899–909, 1999.

[37] M. Riesenhuber, T., and Poggio. Hierarchical Models of Object Recognition in Cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

[38] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Int. Conf. Computer Vision*, pages 636–643, Vancouver, 2001.

[39] B. Schiele and J. Crowley. Recognition without Correspondence using Multidimensional Receptive Field Histograms. *Int. J. Computer Vision*, 36(1):31–50, January 2000.

[40] B. Schiele and A. Pentland. Probabilistic Object Recognition and Localization. In *Int. Conf. Computer Vision*, 1999.

[41] C. Schmid. Weakly supervised learning of visual models and its application to content-based retrieval. *Int. J. Computer Vision*, 56(1):7–16, 2004.

[42] C. Schmid and R. Mohr. Local Grayvalue Invariants for Image Retrieval. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 19(5):530–534, 1997.

[43] T. Serre, L. Wolf, and T. Poggio. Object Recognition with Features Inspired by Visual Cortex. In *Int. Conf. Computer Vision & Pattern Recognition*, 2005.

[44] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

[45] M. Varma and A. Zisserman. Texture Classification: Are filter banks necessary? In *Int. Conf. Computer Vision & Pattern Recognition*, 2003.