INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Hyperfeatures — Multilevel Local Coding for Visual Recognition

Ankur Agarwal — Bill Triggs

N° 5655

August 2005

THEME 3

Rapport de recherche

ISSN 0249-6399

Hyperfeatures —
Multilevel Local Coding for Visual Recognition

Ankur Agarwal* , Bill Triggs †

Theme 3 — Human-computer interaction, image processing, data management, knowledge systems
Project LEAR — Learning and Recognition in Vision

Research Report n° 5655 — August 2005 — 19 pages

Abstract: Histograms of local appearance descriptors are a popular representation for visual recognition. They are highly discriminant and they have good resistance to local occlusions and to geometric and photometric variations, but they are not able to exploit spatial co-occurrence statistics of features at scales larger than their local input patches. We present a new multilevel visual representation, 'hyperfeatures', that is designed to remedy this. The basis of the work is the familiar notion that to detect object parts, in practice it often suffices to detect co-occurrences of more local object fragments – a process that can be formalized as comparison (vector quantization) of image patches against a codebook of known fragments, followed by local aggregation of the resulting codebook membership vectors to detect co-occurrences. This process converts collections of local image descriptor vectors into slightly less local histogram vectors – higher-level but spatially coarser descriptors. Our central observation is that it can therefore be iterated, and that doing so captures and codes ever larger assemblies of object parts and increasingly abstract or 'semantic' image properties. This repeated nonlinear 'folding' is essentially different from that of hierarchical models such as Convolutional Neural Networks and HMAX, being based on repeated comparison to local prototypes and accumulation of co-occurrence statistics rather than on repeated convolution and rectification. We formulate the hyperfeatures model and study its performance under several different image coding methods including clustering based Vector Quantization, Gaussian Mixtures, and combinations of these with Latent Discriminant Analysis. We find that the resulting high-level features provide improved performance in several object image and texture image classification tasks.

Key-words: Computer vision, Visual recognition, Image coding, Image classification

* GRAVIR and INRIA Rhône Alpes, *Email: Ankur.Agarwal@inrialpes.fr*
† GRAVIR and CNRS, *Email: Bill.Triggs@inrialpes.fr*

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Hyperfeatures —
une representation hierarchique locale pour la reconnaissance visuelle

Résumé : Caractériser le contenu d'images de façon robuste et discriminante reste un défi majeur pour la reconnaissance visuelle. Une approche prometteuse consiste à évaluer un jeu de descripteurs visuels locaux invariants sur un ensemble de régions extraites de l'image, et de caractériser leur statistique – et donc le contenu de l'image – par biais de leur histogramme de valeurs quantifiés. Cette méthode « sac de descripteurs » décrit bien l'apparence locale et elle résiste aussi aux occultations et aux déformations géométriques et photométriques, mais elle a du mal à encoder la structure géométrique du scène. Nous présentons une nouvelle représentation visuelle, les « hyperfeatures », qui remédient ce défaut. L'idée de base est de coder non seulement la distribution marginale de classes d'apparence des régions, mais aussi leurs co-occurrences locales, et ceci a plusieurs reprises. À partir des vecteurs de descripteurs locaux de chaque région, la méthode les quantifie et les cumule sur des super-régions locales afin de créer un histogramme d'apparences local pour chaque super-région. Chaque histogramme étant de nouveau un vecteur de descripteurs local, ce procède peut être répété plusieurs fois, à chaque reprise créent sur une région plus grande une indice de niveau plus élevé qui code les co-occurrences des « sous-parties de scène » issues du niveau inférieur. On hypothèse que plus le niveau est élevé, plus ces indices deviennent « sémantiques » et caractérisent le contenu haut-niveau de l'image. Nous décrivons le modèle hyperfeatures et étudions ses performances sous plusieurs méthodes de codage de descripteurs : quantification vectorielle, mélanges gaussiennes, et analyse latente discriminante. Nos expériences de classification d'images et de textures et de localisation d'objets, démontrent qu'incorporer les hyperfeatures améliore la performance de la méthode de base.

Mots-clés : Vision par ordinateur, reconnaissance visuelle, caractérisation et classification d'images

Contents

1 Introduction

Local coding of image appearance based on invariant descriptors is a popular strategy for visual recognition and image classification [6,7,24]. The image is treated as a loose collection of quasi-independent local patches, robust visual descriptors are extracted from these, and some form of aggregation or summarization is used to characterize the statistics of the resulting set of descriptor vectors and hence quantify the image appearance. There are many variants. Patches can be selected at one or at many scales, and either densely, at random, or sparsely according to some local informativeness criterion [8,12,20]. There are many kinds of local descriptors, and they can incorporate various degrees of resistance to common perturbations such as viewpoint changes, geometric deformations, and photometric transformations [16,18,19,27]. Further, summarization can be done in different ways, either over local regions to make a higher-level local descriptor, or globally to make a whole-image descriptor.

A simple example is the 'texton' or 'bag-of-features' method where local image patches or patch feature vectors from the image or local region are coded using vector quantization against a fixed codebook, and the resulting 'votes' for each codebook centre are tallied to produce a histogram characterizing the distribution of patches over the image or local region. The codebooks are typically constructed by running a clustering algorithm such as K-Means over a large set 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 model in descriptor space, with each patch being 'coded' by its vector of posterior mixture-component membership probabilities or latent variable values.

Such local coding approaches were initially developed to characterize image textures [17,22], however it has recently become clear that they provide surprisingly good performance in many object recognition tasks [6,7]. Their main limitation 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), whereas higher order and inter-patch statistics – most notably patch co-occurrences – are important in many recognition tasks. To palliate this, several authors incorporate an additional level of coding that captures pairwise or neighbourhood co-occurrences of coded patches [14,25].

1.1 Hyperfeatures

This paper generalizes and formalizes the above process to higher levels of coding. 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. At each level, the image is divided into local regions with each region being characterized by a descriptor vector. Each vector is produced by coding (*e.g.* vector quantizing) and pooling a local set of 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 on a (typically coarser) grid, so the resulting upper level output is again a grid of descriptor vectors (code 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 histogram vector. The codebooks are learned in the usual way, using the descriptor vectors of the corresponding level from a set of training images. To ensure scale-invariant recognition, the whole process also runs at each layer of a multi-scale image pyramid, so there is actually a pyramid and not a grid of descriptor vectors at each hyperfeature level[1].

We call the resulting higher-level features 'hyperfeatures'. Our main claim is that hyperfeature based coding offers an efficient and natural framework 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 only nearby co-occurrences rather than 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 bag-of-features representations. The basic intuition is that despite their geometric weakness, simple co-occurrences of characteristic object fragments are often *in practice* sufficient cues to deduce the presence of larger object 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. Owing to their loose, agglomerative nature, hyperfeature stacks are naturally robust to occlusions and feature extraction failures – the learning process ensures that common errors are seen and accounted for, and even if the top level object is not coded successfully, substantial parts of it may be, allowing the system to cue off of these. The process of constructing hyperfeatures is illustrated in figure 2.

[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 'quantize-and-histogram' local coding that have been applied, with each transformation producing a different, higher-level 'image' or 'pyramid'. Each fold typically also makes the representation spatially coarser, but this is somewhat incidental.

(a) (b)

Figure 1: Some typical images from the (a) PASCAL object dataset [2], and (b) the KTH-TIPS texture dataset [4] that are used to evaluate hyperfeature based coding for image classification.

1.2 Previous Work

The hyperfeatures representation has several precursors. Classical ‘texton’ or ‘bag of features’ representations are global histograms over quantized image descriptors. This is ‘level 0’ of the hyperfeatures representation[2]. Histograms of quantized level 1 features have been used for texture classification and for recognizing textured objects, *e.g.* by [22, 26], but we are not aware of any previous use of higher-level hyperfeatures.

Hyperfeature stacks also have analogies to multilevel neural models such as Convolutional Neural Networks (CNN) [15] and HMAX [23]. Both CNN and HMAX are multilayer networks with alternating stages of linear filtering (banks of learned convolution filters for CNN’s and of learned ‘simple cells’ for HMAX) and nonlinear rectify-and-pool operations. In CNN’s 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. HMAX lays claims to biological plausibility whereas CNN is more of an engineering solution, but both are convolution based and both are typically trained discriminatively. In contrast the hyperfeatures model is essentially based on making local comparisons with exemplars and accumulating their co-occurrence statistics. Its basic nonlinearity is the ‘similar template’ lookup, or more generally the coding of feature vectors as vectors of membership probabilities of latent classes or mixture components. Pooling is achieved by the sum-and-normalize process used to build local histograms, but this is not really a conventional linear operation in the sense that it is a strictly positive representation that is specifically designed to capture the spatial co-occurrence statistics of different exemplars.

2 Base Features and Image Coding

In this section we discuss the image coding schemes to which we have applied the hyperfeature algorithm. The algorithm itself is designed to be generic and may be used to extend a wide class of image coding techniques to multi-level codings. Here, to understand the relative benefits of higher level coding as compared to more sophisticated uni-level codings, it is applied to a few different coding techniques of varying complexities. For simplicity, the coding methods are described in context of the base level (level 0) in this section, and the generalization to higher levels is left to the next section.

The features used at the base level consist of raw local image features and are referred to as ‘base features’. In our case, they are SIFT-like gradient orientation histograms computed over a dense grid (actually a multiscale pyramid) of image points. The input to the multilevel coding algorithm is thus a dense pyramid of 128-D SIFT descriptor vectors. We compute these in a manner very similar to [16], but using a more robust normalization that is more resistant to image noise in nearly empty patches[3]. Normalization provides resistance to photometric transformations, and the spatial quantization within SIFT allows for reasonable robustness to shifts of the patches by a pixel or two.

There are also several sparser representations such as those that use features extracted at interest points to summarize contents of an image [6, 7, 14, 19], but we have chosen not to study them here as we wanted to measure the basic hyperfeatures performance independent of the details of keypoint detection. The hyperfeature framework also applies to these. Dense representations have recently been explored for visual representations, *e.g.* in [11], and differ from keypoint detection based ones mainly in that they encode information in the complete image and not just that contained around a set of salient points.

[2] Note that the ‘level 1’ hyperfeature vectors are local histograms of the same quantized ‘level 0’ input features that are used to generate the level 0 output. More generally, the information histogrammed globally for level l *output* histograms is also histogrammed locally for level $l+1$ hyperfeatures.

[3] This need arises as the original SIFT computation method was not designed to handle descriptions of non-informative points.

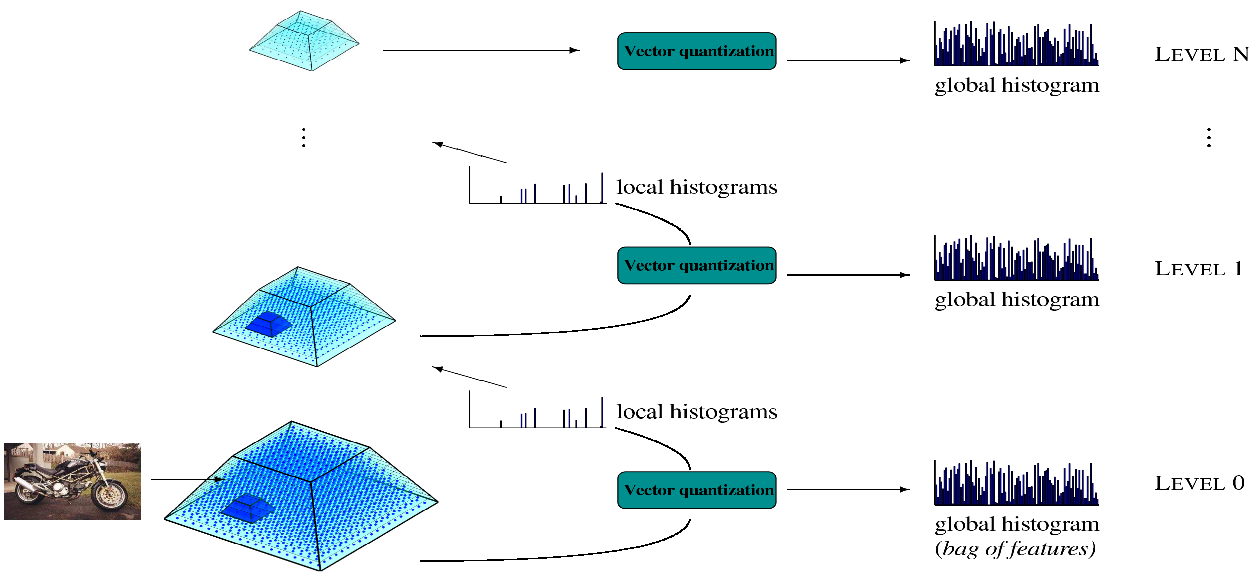


Figure 2: The construction method for hyperfeatures. The 'level 0' (base feature) pyramid is constructed by calculating a local image descriptor vector for each patch in a multiscale pyramid of overlapping image patches. These vectors are then vector quantized according to the level 0 codebook, and local histograms of codebook memberships are accumulated over local position-scale neighbourhoods (shown by the smaller darkened pyramids) to make the level 1 feature vectors. This is essentially a matter of summing codebook membership vectors over the neighbourhood, so other attribution schemes such as soft voting and more generally coding with posterior membership probabilities of mixture components or similar latent models can also be used. Softer spatial windowing is also possible. This process simply repeats itself at higher levels. The coding that generates the level $l+1$ feature vectors is also used to create level l output vectors – here global histograms over the whole level l pyramid. The collected output features can be fed to a learning machine and used to classify the (global or local) image region.

2.1 Vector Quantization

A very simple and widely-used scheme to summarize the content of collections of image patches is vector quantization (VQ): each patch is coded with a label from a vocabulary that is learned from some training set. The experiments in this paper use a clustering algorithm similar to the mean shift based online clustering method of [11] to obtain vector quantization centers. Each descriptor in the pyramid is replaced by its label and histograms of these label frequencies are constructed over the whole image (or over local regions in the pyramid as we will see in the next section). This simple method turns out to be very effective, but abrupt quantization does cause some aliasing effects, which can be handled by **soft vector quantization**: softly voting into several nearby centers for each descriptor. Taking this one step further, as a second coding method, we can use Expectation-Maximization to learn a Gaussian mixture (GM) distribution on the space of descriptors and code each point by its vector of posterior mixture component membership probabilities. This gives rise to qualitatively very different centers than clustering, as shown in figure 3, and also a more effective coding as we shall see later.

2.2 Latent Dirichlet Allocation

The mixture components or quantization labels can be interpreted as latent variables that summarize the contents of each descriptor. Alternatively, one can capture the cooccurence information at a more "semantic" level by performing a latent semantic analysis [9] on the distribution of image descriptors. The individual descriptors – which we will refer to as "words" by analogy with the text modelling literature where such models are popular – are modelled as being produced from a set of latent "topics", with images being analogous to "documents": contextually consistent arrangements of words belonging to different topics. The particular latent semantic analysis model that we use for the experiments in this paper is Latent Dirichlet Allocation (LDA), described in detail in [1]. LDA allows the coding of images (or documents) by learning a set of latent topics in the form of a word-topic matrix β that expresses the probability for each topic to generate each word, with another latent variable (θ) estimated for each document, that estimates the mixture of topics present in that image/document. This may be understood of as a "theme" of an image (in lines with notation in [13]) that governs

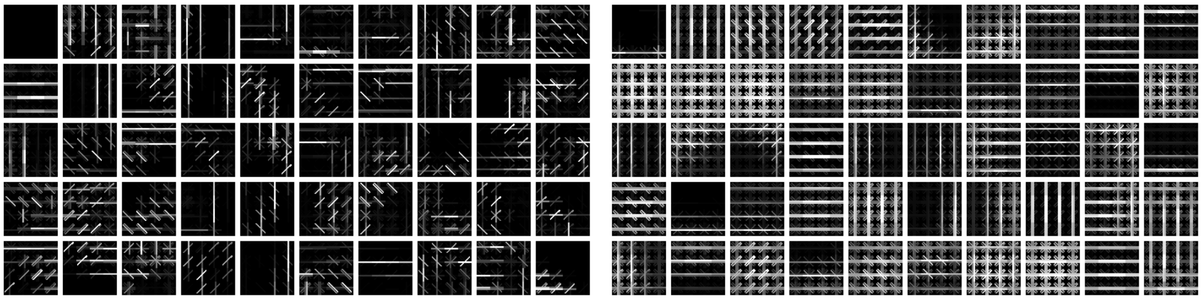


Figure 3: Codebook centers obtained for SIFT descriptors from a dataset of 684 images from 4 object categories. The intensity of each line represents the weight of the corresponding orientation bin in that cell. *(Left)* Vector quantization cluster centers, and *(Right)* Gaussian mixture centers. The two codebooks clearly code information very differently — VQ picks 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. The blank patch occurs very frequently in this particular dataset, mostly from uncluttered parts of background, and is hence almost always prominent amongst the centers.

1. $\forall(i,x,y,s)$, $\mathcal{F}^{(0)}_{ixys} \leftarrow$ 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.
   - $\forall i$:
     - 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).
     - If $l < N$, $\forall(x,y,s)$ 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 $\mathcal{N}^{(l+1)}(x,y,s)$.
3. Return $\{\mathcal{H}^{(l)}_i \,|\, \forall i, l\}$.

Figure 4: The hyperfeature coding algorithm.

what topics are likely to be observed in it. Finally, θ is sampled from a Dirichlet distribution, the parameters α of which are estimated globally from the training data. We typically observe small values of α that intuitively corresponds to each image having a sparse distribution of topics. In the absence of a natural discrete set of "words" for images, we construct a "visual" vocabulary by either clustering and vector quantization, or by using the GM centers. So VQ corresponds to assigning each descriptor to a unique word, while GM can be understood as representing each descriptor as a probability distribution over words. In either case, the statistics of words over either whole images or local image regions can be used to learn an LDA model and infer a set of latent topic activations to represent it.

In the remainder of the paper, we will use the term 'coding' to refer to applying either vector quantization or a Gaussian mixture model, or either of the two followed by LDA summarization.

3 Constructing Hyperfeatures

The process of constructing hyperfeatures is illustrated in figure 2. The co-occurrence statistics of local 'level 0' image features are captured by vector quantizing or otherwise nonlinearly coding the input vectors, and histogramming the results over a moving local region $\mathcal{N}$. This process converts local descriptor vectors (input features) to coarser but higher-level descriptor vectors (local histograms). It is independent of the exact signification of the features used, so it can be repeated recursively to obtain features at multiple levels of abstraction.

We denote the feature pyramids at a particular level l by $\mathcal{F}^{(l)}$. Using (x,y,s) to represent the position-scale coordinates within an image, $\mathcal{F}^{(l)}_{ixys}$ is the level-l descriptor vector at (x,y,s) in image i. During the training phase, a codebook or coding model is learned from all features (all i,x,y,s) at level l. This codebook is then used to code all of the features

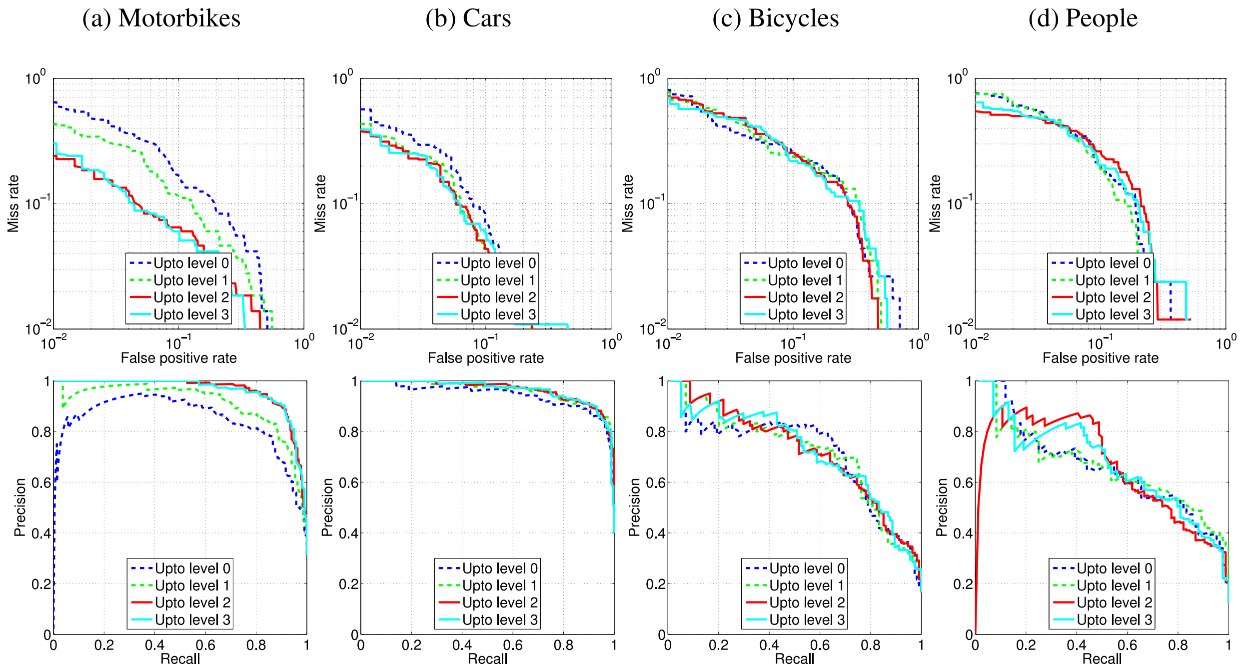


Figure 5: Detection Error Trade-off and Recall-Precision curves for different classes from the objects dataset. Including additional levels of hyperfeatures improves the classification performance up to a certain level. The motorbike, car and bicycle classes achieve best performance at level 3 while the person class performs best with only one level of hyperfeatures. The highest gain on the motorbike and car classes suggests that geometrical structure is captured quite well by hyperfeatures.

at level l in image i, and these are pooled spatially over a local neighbourhood region $\mathcal{N}^{(l+1)}(x, y, s)$ to make $\mathcal{F}^{(l+1)}_{ixys}$. The exact operation depends on the coding method. In vector quantization, this involves a single global clustering followed by local histogramming of class labels within each region; GM involves learning a global mixture model using EM and then summing up membership probabilities for all points within the local region. If LDA is used, its parameters are estimated once over all training images, and then used to infer topic distributions over each region independently, making use of the local context within each region. We also tried applying LDA with global context, *i.e.* treating the whole image as a document, but it turns out to be better to accumulate context locally within each region – *i.e.* we treat each local region as a separate document. In each case, we use $d^{(l)}$ to denote the output feature dimension. The neighbourhood regions are implemented as small trapezoids as shown in figure 2. This shape is motivated by the variation in spatial extent with scale, allowing it to maintain scale invariance, and helping to minimize the boundary effects that cause the pyramids to shrink in size with increasing level. Note that it is essential to change the region size at each level, or at least to use regions that overlap several previous layers – otherwise one is just re-coding the same information (same region) each time. The complete algorithm for a VQ based coding scheme on N levels is summarized in figure 4.

4 Experiments on Image Classification

To illustrate the discriminative capabilities of hyperfeature based coding, we present some image classification experiments on two datasets: a 4-class object dataset used for the European Network of Excellence PASCAL's "Visual Object Classes Challenge" [2] (which in turn is based on the "Caltech 7" and "Graz" datasets [3,21]) and the 10-class KTH-TIPS texture dataset [4]. In total for the 4 categories, the object dataset contains respectively 684 and 689 images in the training and test sets. We scale these to a maximum resolution 320x240 pixels. The texture dataset consists of 450 training and 360 test images over 10 texture classes, mostly 200x200 pixels. As base level features we used the underlying descriptor of Lowe's SIFT method – local histograms of oriented image gradients calculated over 4×4 blocks of 4×4 pixel cells [16][4]. The input pyramid had a scale range of 8:1 with a spacing of $1/3$ octave, and a spatial sampling every 8 pixels at

[4]But note that this is densely tiled over the image without orientation normalization, not used sparsely at feature points and rotated to the dominant local orientation as in [16].

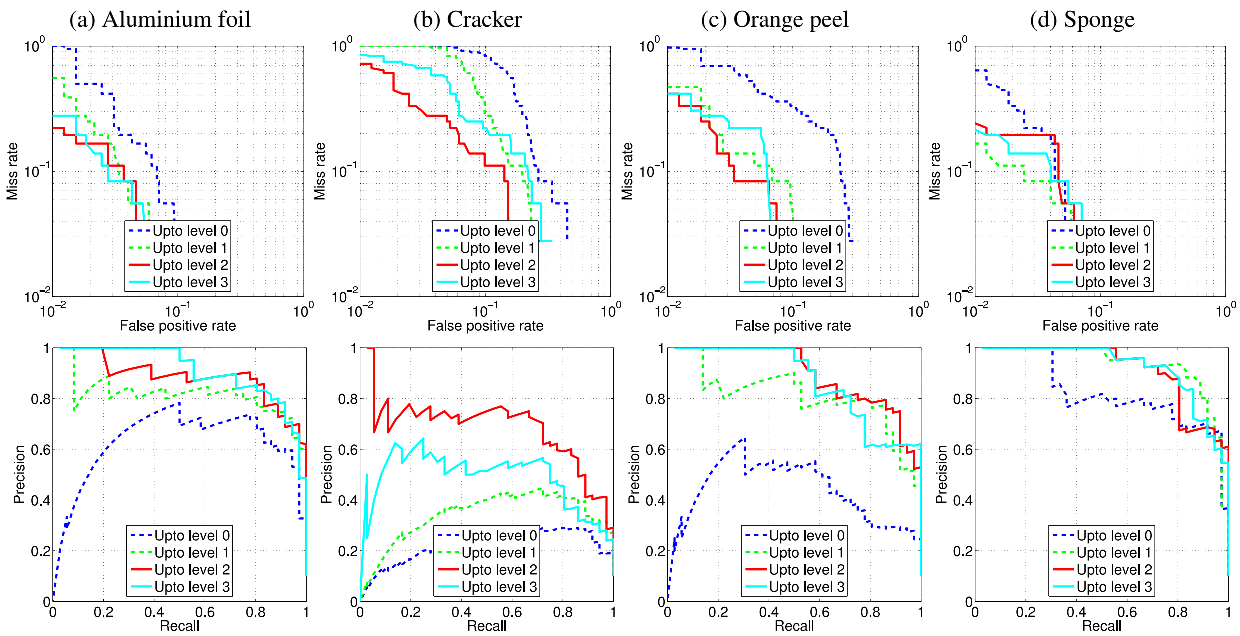


Figure 6: Detection Error Trade-off and Recall-Precision curves for 4 of the 10 classes from the texture dataset, using a mixture of 100 Gaussians at each level. Including additional levels of hyperfeatures improves the classification performance on textures that are poorly classified at level 0. The aluminium and sponge classes show an improvement in classification performance up to level 3, and cracker and orange peel perform their best when using up to level 2.

	Al. foil	Brown bread	Corduroy	Cotton	Cracker	Linen	Orange peel	Sandpaper	Sponge	Styrofoam
A	97.2	88.1	100	86.1	94.4	77.8	94.4	83.3	91.7	88.9
B	100	88.9	100	88.9	91.6	86.1	94.4	83.3	91.7	91.7

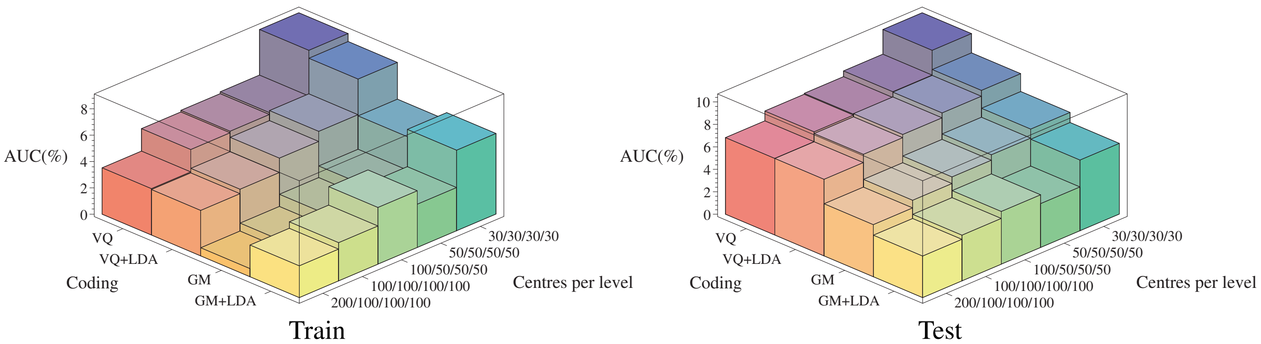
Figure 7: Classification performance (hit rate) at the equal error point for the 10 classes of the KTH-tips texture database, each at its optimal number of levels using (A) hard vector quantization, (B) a diagonal Gaussian mixture model learned by EM.

scale 1, giving a total of 2500-3000 descriptors per image. To construct the neighbourhood regions $\mathcal{N}$, we took 3x3x3 sample volumes in (x, y, s).

The final image classifications were produced by training soft linear one-against-all SVM classifiers over the global-histogram output features collected from the active hyperfeature levels, using SVM-light [10] with standard settings.

Effect of multiple levels: Figure 5 presents DET[5] and precision-recall curves showing the effect of including multiple levels of hyperfeatures in the coding of the PASCAL object dataset using Gaussian mixture coding with a codebook of 200 centers at the base level followed by 100 centers at the higher levels. Including higher levels gives a significant gain for the 'motorbikes' and 'cars' classes but little improvement for 'people'. The results improve up to 3 levels of hyperfeatures, except for 'people' where the results get worse after level 1 and combining base feature histograms with level 1 hyperfeature ones is the best strategy. Presumably the subsequent levels introduce more noise than information in this case. We believe that these differences can be attributed to the varying amounts of *structure* in the different classes. The large appearance variations in the people dataset leave little in the way of regular co-occurrence statistics for the hyperfeature coding to key on, whereas the geometrical structures of cars and motorbikes are modelled well, as is seen in figure 5(a) and (b). Different coding methods and codebook sizes give qualitatively similar behaviour with levels, but the absolute numbers can be quite different as discussed below. Performance curves for a hard vector quantization coding and vector quantization with LDA are given in the appendix. We also studied the effect of changing the size of the pooling neighbourhood regions $\mathcal{N}$ from the default 3x3x3. The coding is found to be not very sensitive to this size.

[5]DET curves plot miss rate *vs.* false positive rate on a log-log scale – the same information as a ROC curve in more visible form. Lower curves are better.



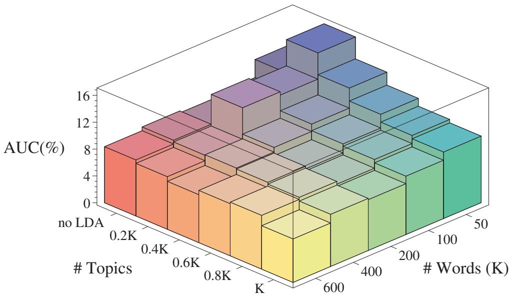
	Train				Test			
# centers at levels 0-1-2-3	VQ	VQ+LDA	GM	GM+LDA	VQ	VQ+LDA	GM	GM+LDA
030-030-030-030	8.94	8.34	5.55	6.11	10.49	8.78	7.19	6.29
050-050-050-050	5.48	5.64	3.02	3.28	8.29	8.26	6.39	3.90
100-050-050-050	5.34	4.88	2.14	4.30	7.78	7.83	5.91	4.69
100-100-100-100	5.22	3.79	1.22	2.90	7.70	7.55	5.32	4.07
200-100-100-100	3.52	3.45	0.62	2.41	6.82	6.82	4.64	3.70
200-200-200-200	3.36	–	0.53	–	6.55	–	4.71	–
500-100-100-100	2.88	–	0.70	–	6.73	–	5.75	–

Figure 8: Average miss rates for the PASCAL objects testset for different codebook sizes and different coding methods. The performance consistently increases as codebook sizes are increased, irrespective of the method used. EM-based Gaussian mixture coding outperforms hard vector quantization even with a significantly smaller number of samples, and performing LDA on top of VQ or GM further improves results.

On the texture dataset, we adopted a slightly different coding mechanism for the base level. Most of the textures are very homogeneous and we find that hard coding using vector quantization tends to assign many of the votes to a ‘default’ center corresponding to a ‘uniform noise’ patch. This noticeably weakens the base level coding and causes performance to drop at all levels – hyperfeatures cannot counter an uninformative base-level coding that assigns all votes to a single centre, as this leaves them with nothing to key on. Stop word removal (deleting the rogue centre and allowing points to assign themselves to the next nearest centre) turns out to be an effective palliative for this, at least in the experiments shown here. Using a Gaussian mixture coding removes the problem and also gives better performance. Results on some classes for this coding are shown in figure 6 (those for the rest of the classes are shown in the appendix). Different texture classes respond differently to higher levels. In four of the ten classes, the performance saturates already at the base level (mostly because almost perfect classification is already attained at that point). The remaining six show gains with up to 3 levels of hyper-coding. The classification performance (hit rate) at the equal error point for VQ and GM on the KTH-TIPS texture dataset is shown in figure 7. In terms of average performance at the optimal level for each class, the method provides a mean hit rate of 91.7% at the equal error point – a slight improvement relative to the bank of filters based approach described in [5], which reports an average performance of 90.6% on the same dataset.

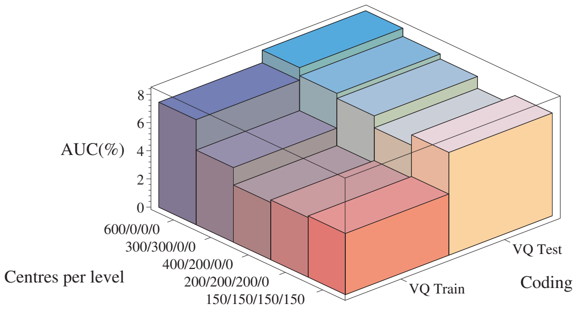
Coding methods and hyperfeatures: The absolute performance is highly dependent on the coding used at each level. One simple observation is that performance improves with increasing size of the codebook (number of clusters / mixture components / latent topics). This is evident in figure 8, which shows the average miss rates[6] for different numbers of centers on different coding methods. For all of the coding methods, adding centers to any of the different levels (going down the columns) leads to a gain in performance. However, some of the methods, notably the Gaussian mixture and LDA, become computationally rather intensive during the learning phase if the codebook is too large. Amongst the different methods, the Gaussian mixture consistently outperforms vector quantization. We attribute this to its *smoother* coding that avoids possible aliasing effects in VQ. Performing LDA on top of either VQ or GM improves the results. In particular, following a Gaussian mixture coding by LDA significantly reduces overfitting. All of the experiments involving LDA in this table use the same number of topics as words and hence do not change the dimensionality.

[6]Average miss rate $= 100 \times (1 - \text{Area Under ROC Curve})$



	# Topics					
# words (K)	no LDA	0.2K	0.4K	0.6K	0.8K	K
50	11.82	16.75	13.49	11.56	10.40	9.90
100	9.13	12.85	10.51	9.53	9.19	8.69
200	8.62	13.34	9.41	8.19	7.90	7.57
400	8.91	8.65	8.72	8.57	8.31	7.76
600	8.35	7.84	7.14	7.77	8.00	6.47

Figure 9: The effect of LDA on average classification performance on test images: average miss rates for the PASCAL objects testset. 'Projecting' words into topic space improves discrimination except for very small numbers of topics over small vocabularies. The performance always improves as the vocabulary size is increased. The first column with no topics refers to using the word counts directly *i.e.* vector quantization with no LDA.



# 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 10: The effect of distributing a fixed number of vector quantization centers across 4 levels: average miss rate for the test object dataset. Encoding the information over multiple levels is helpful, but using too many levels without sufficient coding at each level eventually reduces performance.

In figure 9, we see the effect of varying the number of topics in LDA. A vocabulary of words is built by clustering the descriptor set and word-count histograms are built using simple vector quantization. The advantages of using latent topics as opposed to raw words are clear and again, larger numbers of topics provide more effective coding. For smaller numbers of words, topics equal to 60% of the number of words in number achieve the same performance as simple vector quantization, whereas with increasing size of vocabularies, even 20% perform better.

Given the variation of performance with coding complexity, we also studied the effect of hyperfeature levels for a 'fixed' coding complexity. For this, we fix the total combined number of centers in all the levels of the codebooks and distribute these centres differently across different levels. Figure 10 shows that having more centers on higher levels provides a more effective coding, confirming the importance of higher levels of abstraction. For 600 centers, we see that using up to 3 levels of hypercoding consistently improves the discrimination ability, but beyond this the performance drops – possibly due to the insufficient numbers of centers at the lower levels in this case.

5 Object Localization

One advantage of hyperfeatures is that they capture local image information at several different levels of abstraction. There is a tradeoff 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, even quite high-level hyperfeatures are still local enough to provide good region-level image labelling. Here we use this in a completely bottom-up manner, localizing objects of interest by dividing the image into local regions, in each region building 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) and using the resulting region descriptors to learn local region-level classifiers for each class. In this paper our goal is to study the representational powers of hyperfeatures not frameworks for object recognition, so the experiments below classify regions individually without any attempt to include top-down or spatial contiguity information.

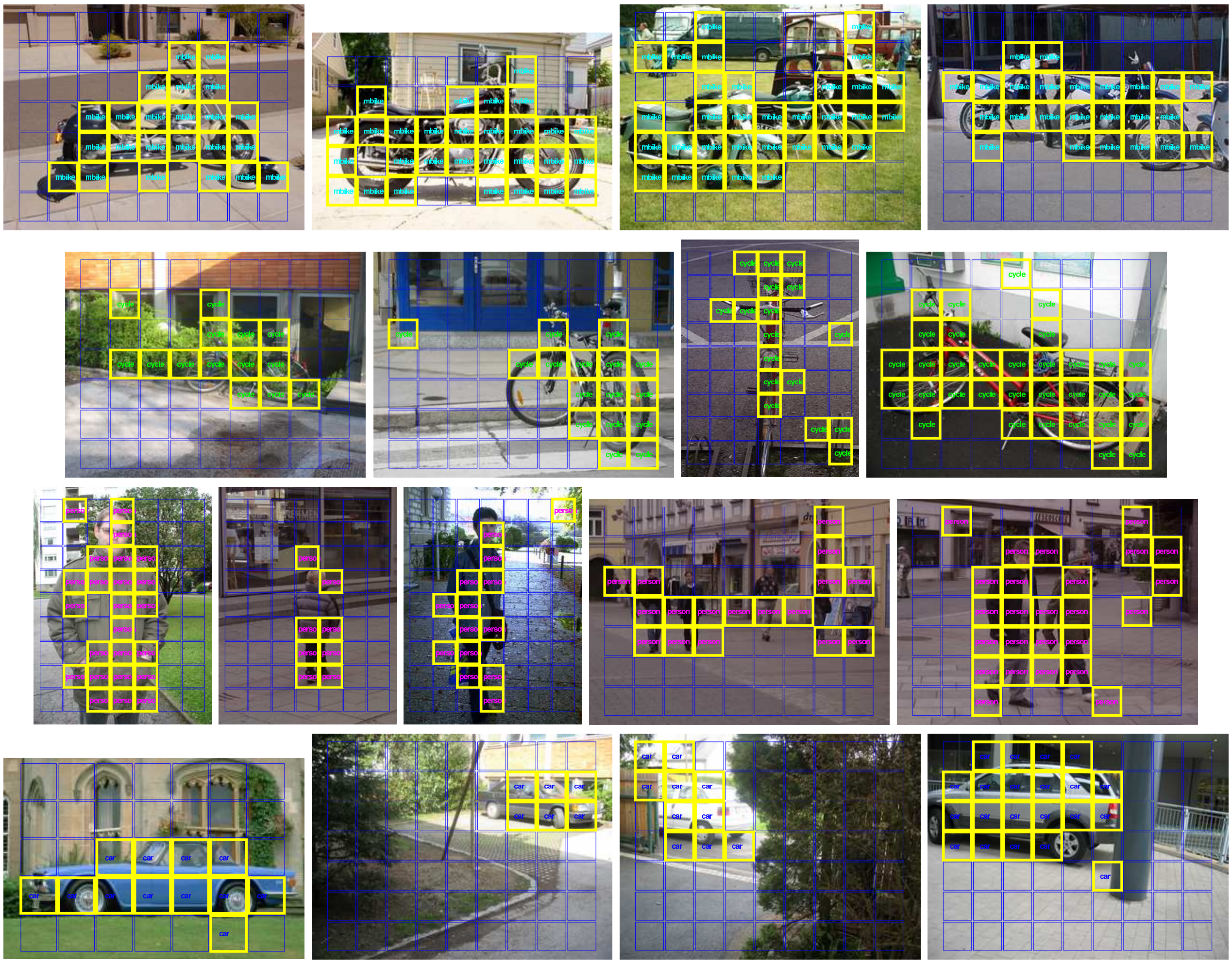


Figure 11: Object localization by classifying local image regions using hyperfeatures. PASCAL dataset. Each row shows examples of results using one of the four independent classifiers, each being trained to classify foreground regions of its own class against the combined set of all other regions – background regions and foregrounds from other classes. An image region is labelled as belonging to the object class if the corresponding SVM returns a positive score.

However, as with other classes of local features, such information is important for classification and should be included in any practical recognition system.

Training with labelled regions. Our first experiment used a fairly strong labelling of training regions. For each object class, all regions from all images of the training set were included, with regions being considered positive if and only if they overlay the mask of an object of that class. Using these labellings, separate linear SVM classifiers over the region's hyperfeature histograms were trained for each class.

We used the rectangular object masks provided with the PASCAL dataset. Note that bounding rectangles are necessarily a rather loose fit for many of the training objects, so for each class a substantial number of pure background patches have effectively been labelled as foreground. Also, objects of one class sometimes occur (unlabelled) in the backgrounds of other classes, and, *e.g.*, instances of people sitting on motorbikes are labelled as motorbike regions not person ones. These imperfections visibly lead to some 'leakage' of labels below, which we would expect a more consistent labelling scheme to remove.

The four classifiers were then used to label local regions of test images. Although the individual patches may seem to contain relatively little discriminant information, we find that they are often labelled correctly, allowing objects to be loosely localized in the images. Figure 11 shows some results based on using the one-against-all classifiers individually. The average accuracy in classifying local regions 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. Again hyperfeatures prove to have more discriminative power than the base features alone, giving an average gain of

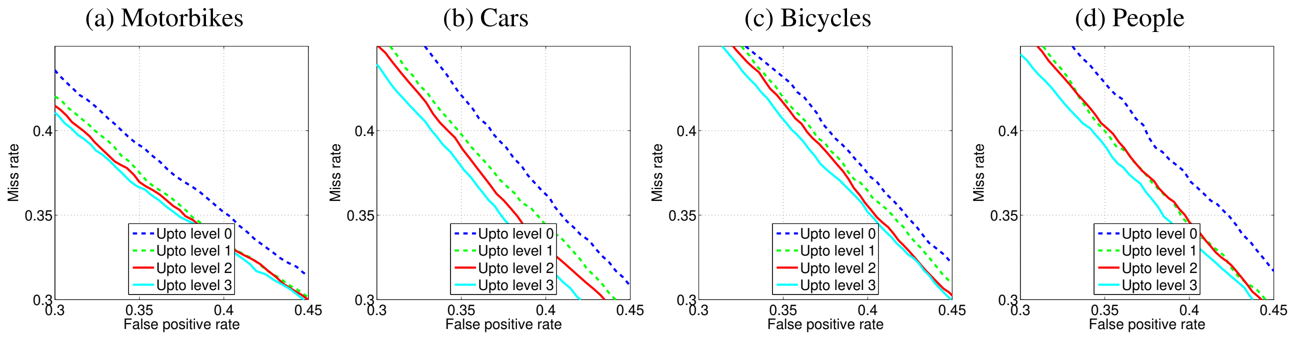


Figure 12: Detection Error Trade-off curves for local region classification on the objects test dataset, showing that including several levels of hyperfeatures gives a clear advantage in discriminating foreground regions from background.

true \ est.	mbike	cycle	person	car
mbike	**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**
bkground	16.36	19.81	19.46	23.46
negative	**22.98**	**25.81**	**19.74**	**25.07**

(a)

true \ est.	mbike	cycle	person	car	bkground
mbike	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
bkground	7.48	13.66	15.99	19.09	43.78
true prop.	20.62	9.50	3.52	4.71	61.65

(b)

Figure 13: (a) Entries from the four two-class confusion matrices, showing the percentage of instances of each true class that are estimated as positive by each of the four classifiers. Each classifier treats regions from all other classes and from the background as 'negatives' (different for each classifier). Labelling is based on the sign of the SVM output. The classifiers are run independently so the classifications are not mutually exclusive. (b) Confusion matrix obtained by running all 4 classifiers in parallel and labelling each region as exactly one of the 4 classes (according to the largest SVM output) or as background (if all outputs are negative). The last row shows the true proportions of the classes in the test set, *i.e.* the performance of a random labelling.

4–5% in classification performance. The effect of multilevel coding on performance is demonstrated in figure 12. The key entries of the four two-class confusion matrices are shown in figure 13(a), with negatives being further broken down into background patches and patches from the three remaining classes. Figure 13(b) shows the multi-class confusion matrix obtained using the maximum of the four classifier outputs to label each region as belonging to just one of the classes. Many of these errors are due to the inexact labelling of the training regions – for example road patches tends to be labelled 'car' and grass ones 'bicycle' as these backgrounds are common in their respective classes – but others correspond to genuine confusion among classes. Some examples of multi-class labellings are shown in figure 14.

Training with labelled images. We also tested a slightly less supervised method based on image-level classification without explicit object/background separation masks. In this, all regions of motorbike images were labelled as positives when training the motorbike classifier, which thus learned to identify "regions of motorbike images" rather than motorbikes per se. Ideally such classifiers would learn that background features were noise because they appear in all classes, but in practice they tend to pick up on small differences in the backgrounds of the different classes – both meaningful contextual ones and random unintentional ones – and use these for discrimination. Also, the classification task becomes more difficult because it now includes (more) difficult to classify background patches. These effects reduce the performance to an average (over all four one-against-all classifiers) hit rate of around 65%. Nevertheless, the foreground object parts typically receive higher weighting than background patches owing to their greater discriminative power, and objects are loosely 'segmented' without using any prior segmentation knowledge during training. As might be expected, contextual associations are common in these experiments. For example road regions are often classified as 'car' because they occurred most often in the training images of cars. Some examples are shown in figure 14(b).

We are currently working on improving the object localization capabilities of the model. On the one hand, spatial coherency in patch labels will be exploited using a Markov Random Field type of model. On the other we are trying

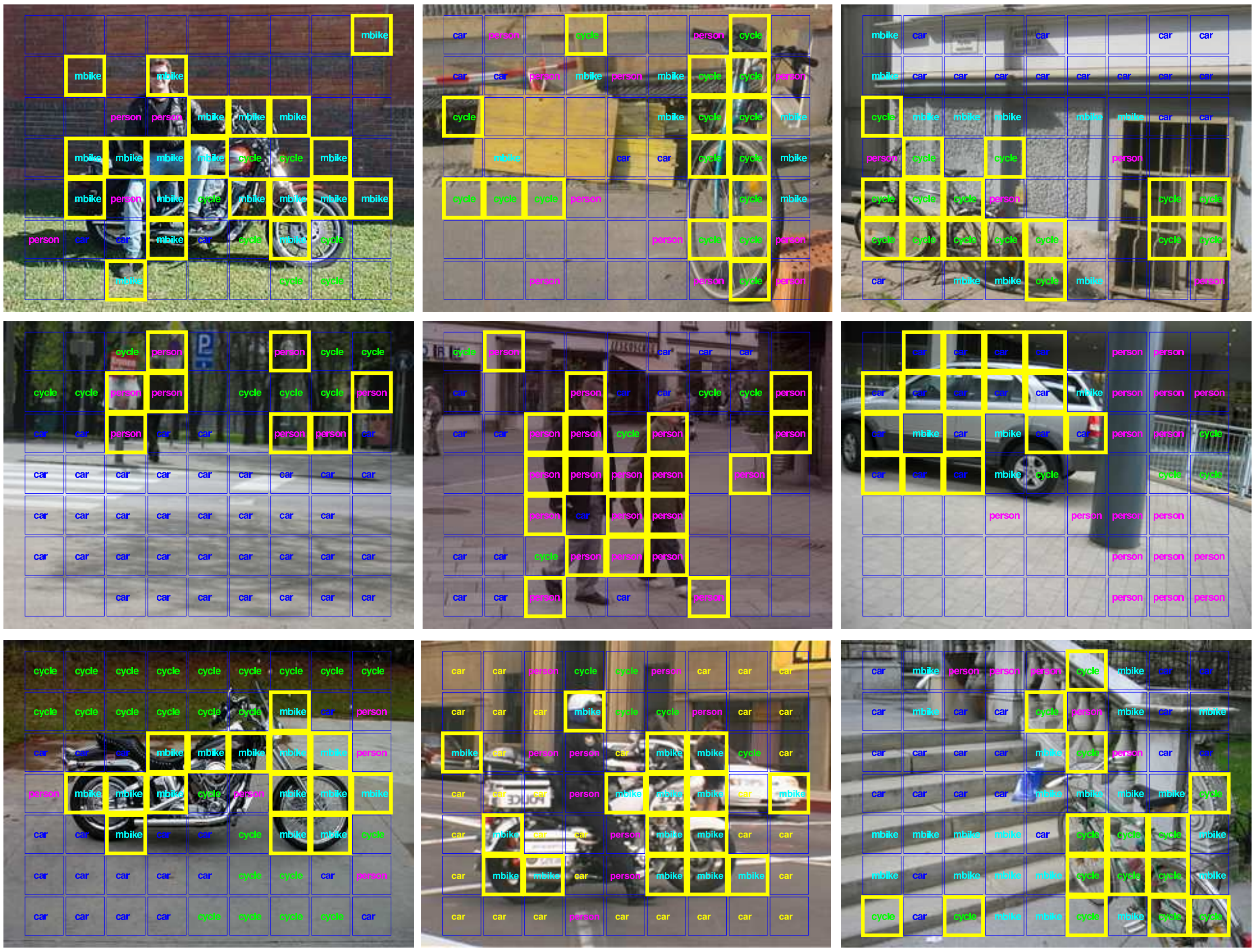


Figure 14: Object recognition and localization by classifying local image regions using hyperfeatures. (a) Top two rows: the classifiers are trained on local regions using approximate foreground/background labellings (in the form of bounding boxes marked around each object in the training images) and each region is labelled as a motorbike/cycle/person/car or as background. (b) Bottom row: the classifiers are trained using image labels alone, without segmentation information, so the labels denote 'region from a car image' not 'region on a car', *etc*. Contextual associations are evident, *e.g*. road regions are often classified as 'car' because they were mostly seen in images from the car category and grass is sometimes associated with bicycles. The second image illustrates that multiple objects can be segregated – this image was labelled as a motorbike in the dataset but instances of person are correctly identified on it.

to learn more focused codebooks by including priors that encourage object-specific groups of labellings and explicitly allowing for a (non-discriminative) background class while training.

6 Conclusions and Future Work

We have introduced a new multilevel nonlinear image coding mechanism called hyperfeatures, 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. Our experiments show that the introduction of one or more levels of hyperfeatures improves performance in many classification tasks, especially when there is discriminative higher-level object structure that can be captured.

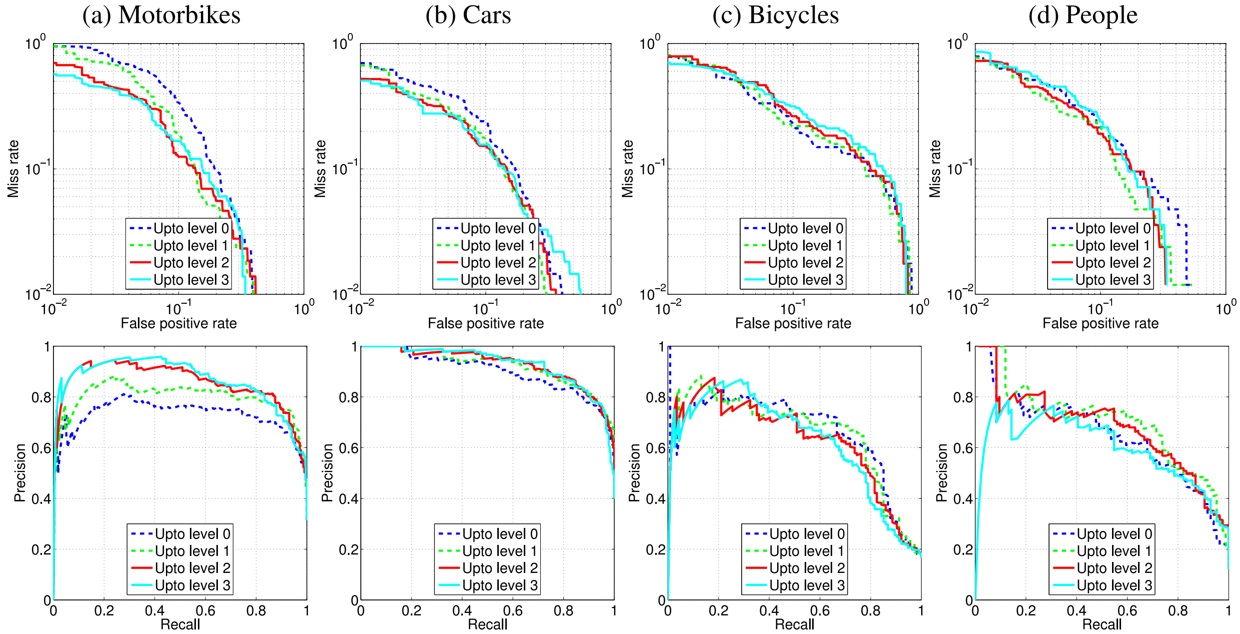
Future work: The hyperfeatures idea is applicable to a wide range of problems involving part-based representations. In this paper the hyperfeature codebooks have been trained by unsupervised clustering, but more discriminative training methods should be a fruitful area for future investigation, *e.g*. image class labels could usefully be incorporated into the learning of latent topics. We also plan to investigate more general Latent Dirichlet Allocation like methods that use local context while training. One way to do this is to formally introduce a “region" (or “subdocument") level in the word–topic–document hierarchy. An interesting aspect that such models should allow us to model is contextual information at different levels of support – this may be useful for object detection.

Concerning image coding methods, several questions remain to be addressed. Comparisons between dense grid and sparse keypoint based representations under similar conditions are needed to understand the relative merits of the two approaches. Further, studying how densely the dense methods need to sample in the image is necessary to avoid both redundancy and loss of useful information. The kinds of features that are appropriate for different image classes and how different clustering methods interact with these also needs to be explored in more detail.

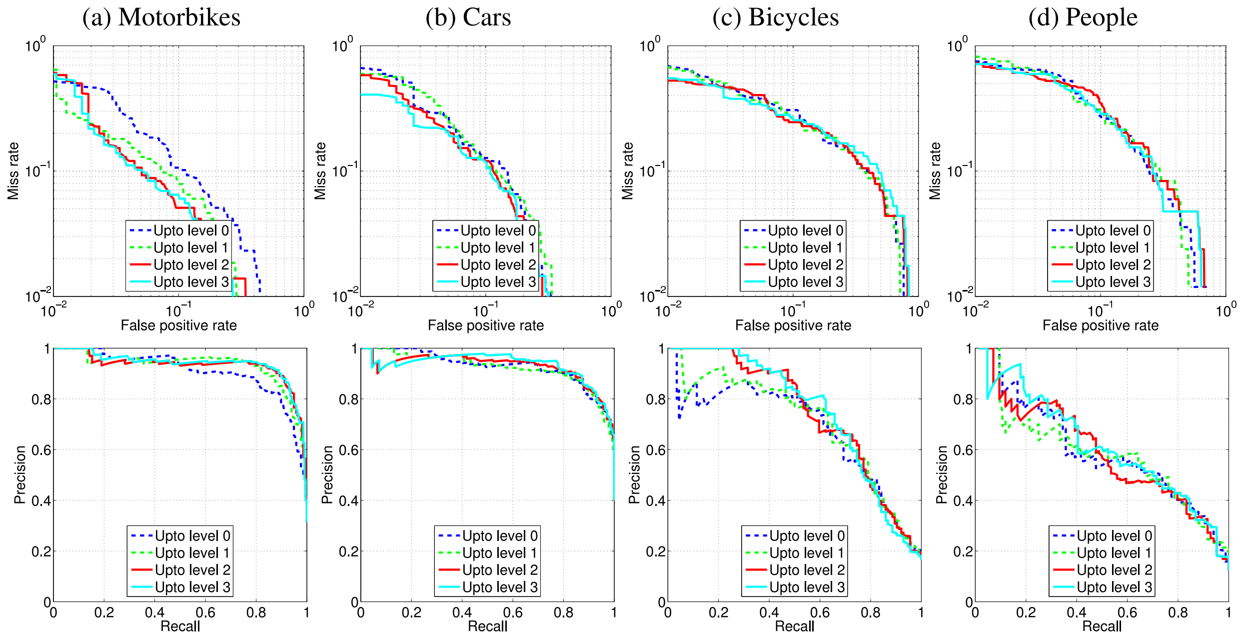
Acknowledgments

We would like to thank the European projects LAVA and PASCAL for financial support, and Diane Larlus, Frederic Jurie, Gyuri Dorkó and Navneet Dalal for comments and code.

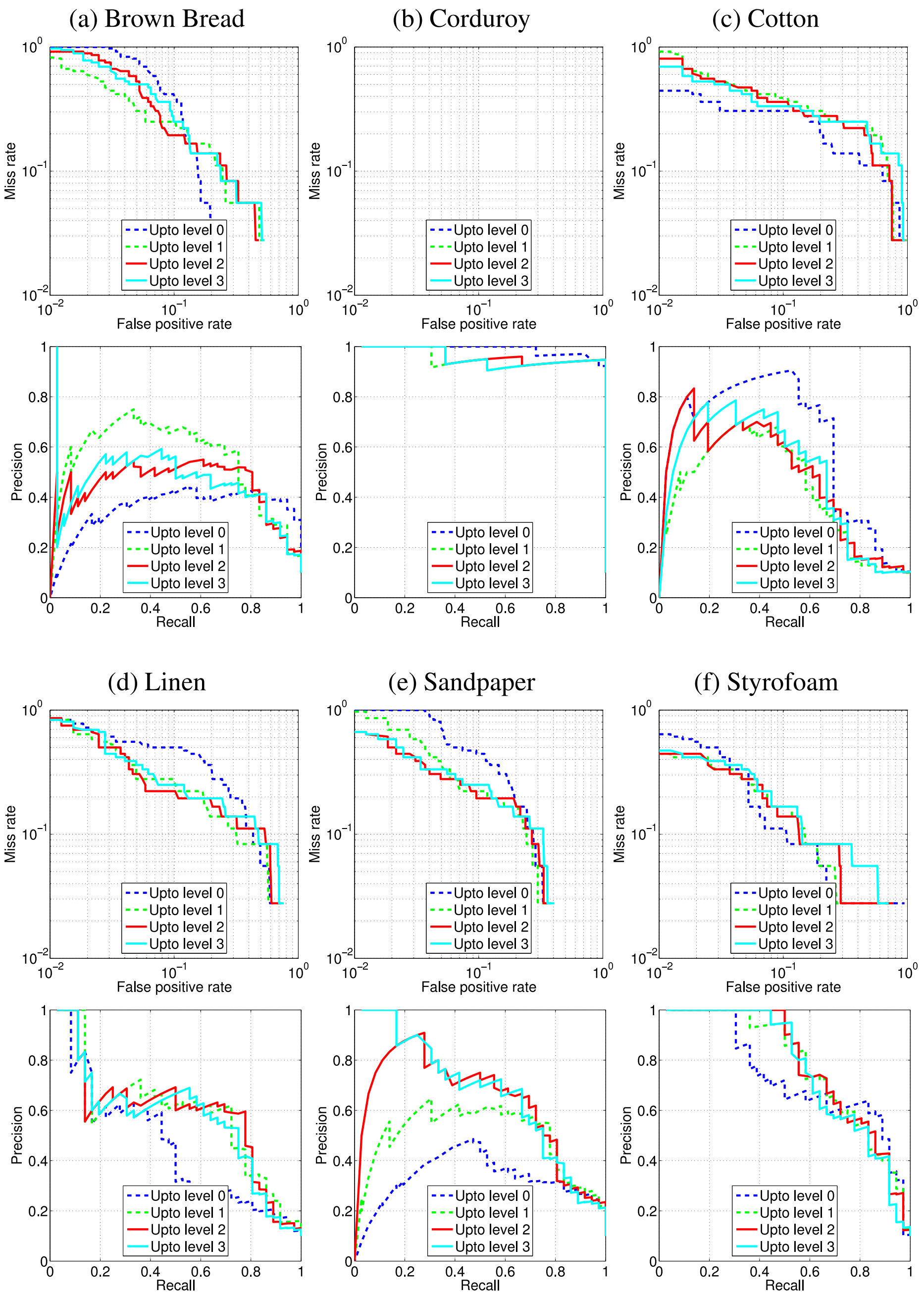
Appendix



Appendix 1: Detection Error Trade-off and Recall-Precision curves for the objects dataset using vector quantization coding. Overall this is slightly worse than the Gaussian mixture coding in figure 5, but the qualitative effect of adding higher levels of hyperfeatures remains the same — the highest gain is achieved on the motorbike and car classes.



Appendix 2: Detection Error Trade-off and Recall-Precision curves for the objects dataset using vector quantization combined with LDA. This coding performs better than VQ alone (appendix 1) on an average. (This is more visible in the Recall-Precision than the DET curves.) The contribution of hyperfeatures remains the same despite the different coding methods, demonstrating the algorithm's applicability to different coding methods.



Appendix 3: Detection Error Trade-off and Recall-Precision curves for the remaining texture classes from figure 6. Additional levels of hyperfeatures give most benefit for the classes that are often confused using only the base features. Corduroy is almost perfectly classified (its DET curve is off the graph) and higher levels actually lead to overfitting in this case.

References

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

[2] Visual Object Classes Challenge. The PASCAL Object Recognition Database Collection. Available at www.pascal-network.org/challenges/VOC.

[3] Rob Fergus and Pietro Perona. The Caltech database. Available at www.vision.caltech.edu/html-files/archive.html.

[4] M. Fritz, E. Hayman, B. Caputo, and J.-O. Eklundh. The KTH-TIPS database. Available at www.nada.kth.se/cvap/databases/kth-tips.

[5] 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.

[6] C. Schmid G. Dorko. Object class recognition using discriminative local features. *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[7] Gabriela Csurka and Cedric Bray and Chris Dance and Lixin Fan. Visual categorization with bags of keypoints. In *European Conf. Computer Vision*, 2004.

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

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

[10] T. Joachims. Making large-Scale SVM Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

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

[12] Timor Kadir and Michael Brady. Saliency, Scale and Image Description. *Int. J. Comput. Vision*, 45(2):83–105, 2001.

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

[14] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Affine-Invariant Local Descriptors and Neighborhood Statistics for Texture Recognition. In *International Conference on Computer Vision*, volume 1, pages 649–655, 2003.

[15] Yann LeCun, Fu-Jie Huang, and Leon Bottou. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *Proceedings of CVPR'04*. IEEE Press, 2004.

[16] D. Lowe. Distinctive Image Features from Scale-invariant Keypoints. *International Journal of Computer Vision*, 60, 2:91–110, 2004.

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

[18] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *To appear in PAMI*, 2005.

[19] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *To appear in International Journal of Computer Vision*, 2005.

[20] H.P. Moravec. Towards Automatic Visual Obstacle Avoidance. In *IJCAI*, page 584, 1977.

[21] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. The Graz image databases. Available at http://www.emt.tugraz.at/∼pinz/data/.

[22] Jan Puzicha, Thomas Hofmann, and Joachim M. Buhmann. Histogram Clustering for Unsupervised Segmentation and Image Retrieval. *Pattern Recognition Letters*, 20:899–909, 1999.

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

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

[25] Bernt Schiele and Alex Pentland. Probabilistic Object Recognition and Localization. In *Int. Conf. Computer Vision*, 1999.

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

[27] Cordelia Schmid and Roger Mohr. Local Grayvalue Invariants for Image Retrieval. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 19(5):530–534, 1997.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399