To appear in Proceedings of the 9th European Conference on Computer Vision, 2006.

Hyperfeatures – Multilevel Local Coding for Visual Recognition

Ankur Agarwal and Bill Triggs

GRAVIR-INRIA-CNRS, 655 Avenue de l'Europe, Montbonnot 38330, France
{Ankur.Agarwal,Bill.Triggs}@inrialpes.fr,
http://www.inrialpes.fr/lear/people/{agarwal,triggs}

Abstract. Histograms of local appearance descriptors are a popular representation for visual recognition. They are highly discriminant and have good resistance to local occlusions and to geometric and photometric variations, but they are not able to exploit spatial co-occurrence statistics at scales larger than their local input patches. We present a new multilevel visual representation, 'hyperfeatures', that is designed to remedy this. The starting point 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 (*e.g.* 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 local collections of image descriptor vectors into somewhat less local histogram vectors – higher-level but spatially coarser descriptors. We observe that as the output is again a local descriptor vector, the process can be iterated, and that doing so captures and codes ever larger assemblies of object parts and increasingly abstract or 'semantic' image properties. 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 Dirichlet Allocation. We find that the resulting high-level features provide improved performance in several object image and texture image classification tasks.

1 Introduction

Local codings of image appearance based on invariant descriptors are a popular representation for visual recognition [40, 39, 3, 30, 12, 26, 27, 11, 36, 22, 13]. The image is treated as a loose collection of quasi-independent local patches, robust visual descriptors are extracted from these, and a statistical summarization or aggregation process is used to capture 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 local informativeness criteria [19, 23]. There are many kinds of local descriptors, which can incorporate various degrees of resistance to common perturbations such as viewpoint changes, geometric deformations, and photometric transformations [43, 30, 39, 32, 33]. Aggregation can be done in different 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.* [31, 29]), but turns out to give surprisingly good performance in many image classification and object recognition tasks [44, 12, 11, 36, 22, 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 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 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 [37, 41, 42, 3, 26].

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 intuition 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 the local histogram model (*e.g.* [37, 42]). 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, using the descriptor vectors of the corresponding level from a set of training images. To promote scale-invariant recognition, the whole process also runs at each layer of a conventional multi-scale image pyramid, so there is actually a pyramid, not a grid of descriptor vectors at each level of the hyperfeature hierarchy[1]. The hyperfeature construction process is illustrated in fig. 1.

[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'.

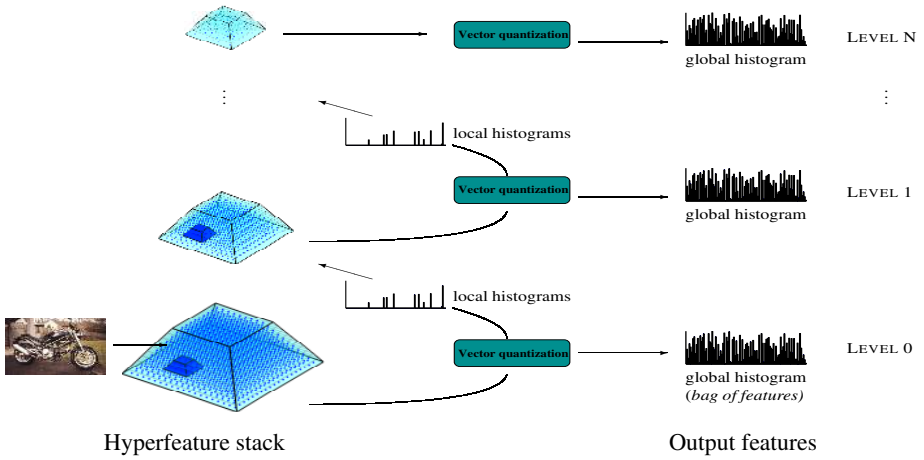


Fig. 1. Constructing a hyperfeature stack. The 'level 0' (base feature) pyramid is constructed by calculating a local image descriptor vector for each patch in a multiscale pyramid of overlapping image patches. These vectors are vector quantized according to the level 0 codebook, and local histograms of codebook memberships are accumulated over local position-scale neighbourhoods (the smaller darkened regions) to make the level 1 feature vectors. 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.

Our main claim is that hyperfeature based coding is a natural feature extraction 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 only codes 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 texton representations. The basic intuition is that despite their geometric weakness, *in practice* simple co-occurrences of characteristic object fragments are often 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. Even when the top level object is not coded successfully, substantial parts of it are captured by the lower levels of the hierarchy and the system can still cue recognition on these.

1.2 Previous Work

The hyperfeature representation has several precursors. Classical 'texton' or 'bag of features' representations are global histograms over quantized image descriptors – 'level 0'

of the hyperfeature representation [31, 29]. Histograms of quantized 'level 1' features have also been used to classify textures and to recognize regularly textured objects [37, 42] and a hierarchical feature-matching framework for simple second level features has been developed [25].

Hyperfeature stacks also have analogies with multilevel neural models such as the neocognitron [18], Convolutional Neural Networks (CNN) [28] and HMAX [38]. 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 atleast one associated lower level cell is active. 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. The neocognitron and HMAX lay claims to biological plausibility whereas CNN is more of an engineering solution, but all are convolution based and typically trained discriminatively. 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 choice of codebook. The basic nonlinearity is also different: exemplar comparison by nearest neighbour lookup – or more generally nonlinear codings based on membership probabilities of latent patch classes – followed by a comparatively linear accumulate-and-normalize process for hyperfeatures, *versus* linear convolution filtering followed by simple rectification for the neural models.

The term 'hyperfeatures' itself has been used to describe combinations of feature position with appearance [14]. This is very different from its meaning here.

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 we have tested so far. For simplicity we describe them in the context of the base level (level 0).

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 [30] but using a normalization that is more resistant 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 is thus a pyramid of 128-D SIFT descriptor vectors. But other descriptors could also be used (*e.g.* [34, 4]).

Hyperfeature models based on sparse (*e.g.* keypoint based [12, 11, 26, 33]) feature sets would also be possible but they are not considered here, in part for simplicity and space reasons and in part because recent work (*e.g.* [22]) 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 [29]. 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 SIFT descriptors, with a vocabulary learned from a training set using a clustering algorithm similar to the mean shift based on-line clusterer of [22]. 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.

Although vector quantization turns out to be very effective, 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. 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) [20] perform better. There are many variants on pLSA, typically adding further layers of latent structure and/or sparsifying priors that ensure crisper distinctions [8, 9, 24, 7]. Here we use **Latent Dirichlet Allocation (LDA)** [5]. 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 from 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 word its own topic allows more variety than sharing a single fixed mixture of topics across all words 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.

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 $l < N$, $\forall(x,y,s)$ 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 $\mathcal{N}^{(l+1)}(x,y,s)$.
     - 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 $\{\mathcal{H}^{(l)}_i \,|\, \forall i,l\}$.

Fig. 2. The hyperfeature coding algorithm.

In our case – both during 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 over which hyperfeature coding is occurring. 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. 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, it can be repeated recursively over higher and higher order neighbourhoods to obtain a series of increasingly high level but spatially coarse descriptor vectors.

Let $\mathcal{F}^{(l)}$ denote the hyperfeature pyramid at level l, (x,y,s) denote position-scale coordinates within a feature pyramid, $d^{(l)}$ denote the feature or codebook/histogram dimension at a level l, 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 $\mathcal{F}^{(l+1)}_{ixys}$. The complete algorithm for VQ coding on N levels is summarized in figure 2.

For vector quantization, coding involves a single global clustering for learning, followed by local histogramming of class labels within each neighbourhood for use. For

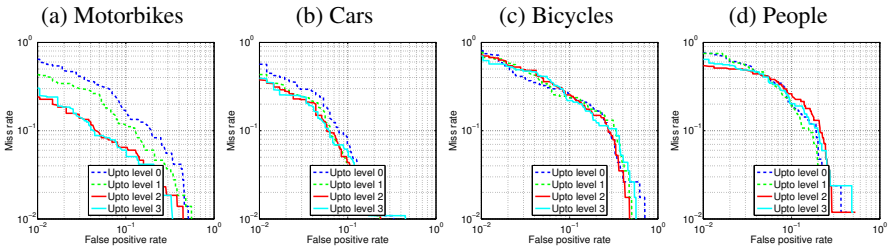


Fig. 3. Detection Error Trade-off curves for the classes of the PASCAL dataset. Up to a certain level, including additional levels of hyperfeatures improves the classification performance. For the motorbike, car and bicycle classes the best performance is at level 3, while for the person class it is at level 1 (one level above the base features). The large gain on the motorbike (a $5\times$ 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.

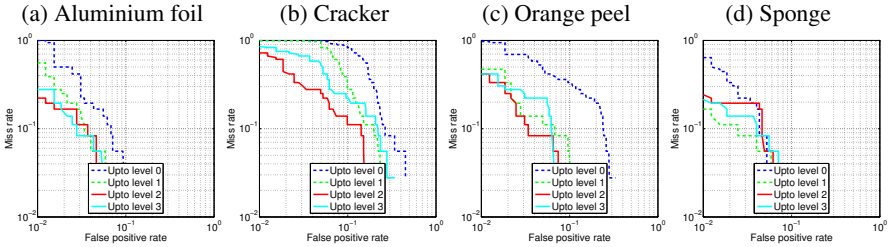
GM, a global mixture model is learned using EM, and in use the mixture component membership probability vectors of the neighbourhood's patches are summed to get the code vector. If LDA is used, its parameters $\alpha, \boldsymbol{\beta}$ are estimated once over all training images, and then used to infer topic distributions over each neighbourhood independently, *i.e.* each neighbourhood is a separate 'document' with its own LDA context.

In all of these schemes, the histogram dimension is the size of the codebook or 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 level is a parameter. The effective region size should grow with the level – otherwise the same information is re-encoded each time, which tends to cause rapid saturation and suboptimal performance.

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" [15] and "Graz" [35] datasets that was used for the European network PASCAL's "Visual Object Classes Challenge" [10]; the 10 class KTH-TIPS texture dataset [16]; and the CRL-IPNP dataset of line sketches used for picture naming in language research [1]. The PASCAL 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 which we divide into two classes, images of people and others. 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 [30][2]. The input pyramid had a scale range of 8:1 with a

[2] But note that this is tiled densely over the image with no orientation normalization, not applied sparsely at keypoints and rotated to the dominant local orientation as in [30].



	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

Fig. 4. *Top:* Detection Error Trade-off curves for 4 of the 10 classes from the KTH-TIPS dataset, using a mixture of 100 Gaussians 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. *Bottom:* One-vs-rest classification performance (hit rate) at the equal error point for the 10 classes of this 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.

spacing of $1/3$ octave and patches sampled at 8 pixel intervals, giving a total of 2500-3000 descriptors per image. For the pooling neighbourhoods $\mathcal{N}$, we took volumes of $3\times3\times3$ 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.

The final image classifications were produced by training soft linear one-against-all SVM classifiers independently for each class over the global output histograms collected from the active hyperfeature levels, using SVM-light [21] with default settings.

Effect of multiple levels: Figure 3 presents DET[3] curves showing the influence of hyperfeature levels on classification performance for the PASCAL 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. We believe that the difference in behaviour between classes can 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 3(a) and (b). Different coding methods and codebook sizes have qualitatively similar evolutions the absolute numbers can be quite different (see below).

[3] 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 values are better.

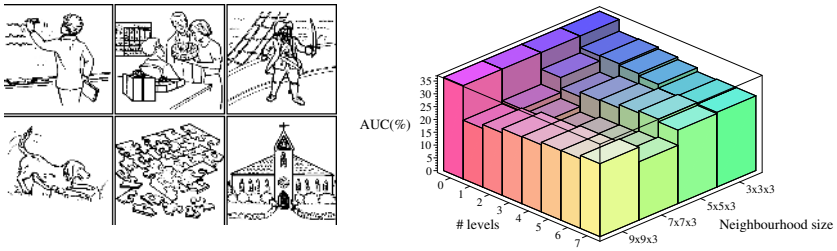


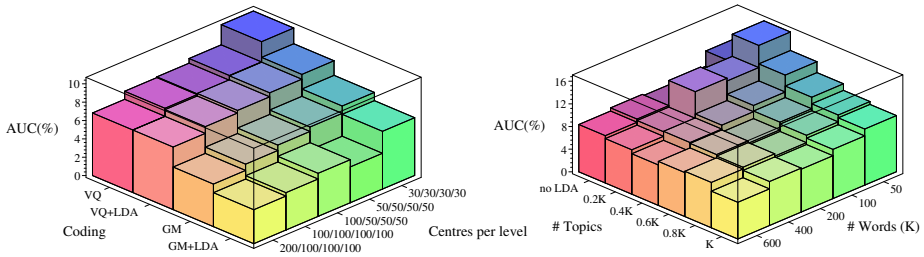
Fig. 5. *Left:* Sample positive (people) and negative (object/scene) pictures from the CRL-IPNP dataset. *Right:* Average miss rates on the positive class for different pooling neighbourhood sizes and different numbers of hyperfeature levels. For a 3x3x3 neighbourhood (in x, y, s), 5 levels of hyperfeatures are best, but the best overall performance is achieved by 7x7x3 neighbourhoods with 3 levels of hyperfeatures.

The results on the KTH-TIPS texture dataset in fig. 4 (top) lead to similar conclusions. For 4 of the 10 classes the level 0 performance is already near perfect and adding hyperfeatures makes little difference, while for the remaining 6 there are gains (often substantial ones) up to hyperfeature level 3. The texture classification performance at equal error rates for VQ[4] and GM coding is shown in fig. 4 (bottom). GM is better on average. Overall, its mean hit rate of 91.7% at equal error is slightly better than the 90.6% achieved by the bank of filters approach in [17] – a good result considering that in these experiments relatively few centres, widely spaced samples and only a linear SVM were used. (Performance improves systematically with each of these factors).

On the CRL-IPNP dataset, we find that 4 or 5 levels of hyperfeatures give the best performance, depending on the size of the pooling regions used. See fig. 5.

Coding methods and hyperfeatures: Fig. 6 (left half) shows average miss rates (1 − Area Under ROC Curve) on the PASCAL 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 no 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 crisply than VQ or GM, presumably because it codes them by selecting a sparse palette of topics rather than an arbitrary vector of

[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. Certain textures are homogeneous enough to cause all bins of the SIFT descriptor to fire about equally, giving 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	–

# centers, 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

Fig. 6. Average miss rates on the PASCAL objects test set. *Left (plot and table):* Miss rates for different codebook sizes and coding methods. Larger codebooks always give better performance. GM coding outperforms VQ coding even with significantly fewer centres, and adding LDA consistently improves the results. The LDA experiments 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. *Top right:* For the LDA method, performance improves systematically as both code centres (here VQ) and LDA topics are added. *Bottom right:* For a fixed total number of centers (here VQ ones), performance improves if they are distributed relatively evenly across several levels (here 3 levels, with the inclusion of a 4^{th} reducing the performance): adding higher level information is more useful than adding finer-grained low level information.

codes. If used to reduce dimensionality, LDA may also help simply by reducing noise or overfitting associated with large VQ or GM codebooks, but this can not be the whole story as LDA performance continues to improve even when there are more topics than input centres. (*c.f.* fig. 6 top right.)

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. Fig. 6 (bottom 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.

5 Object Localization

One advantage of hyperfeatures is that they offer a controllable 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 useful object-region level image labeling. Here we use this for bottom-up localization of possible objects of interest. The image pyramid is tiled with regions and in each region

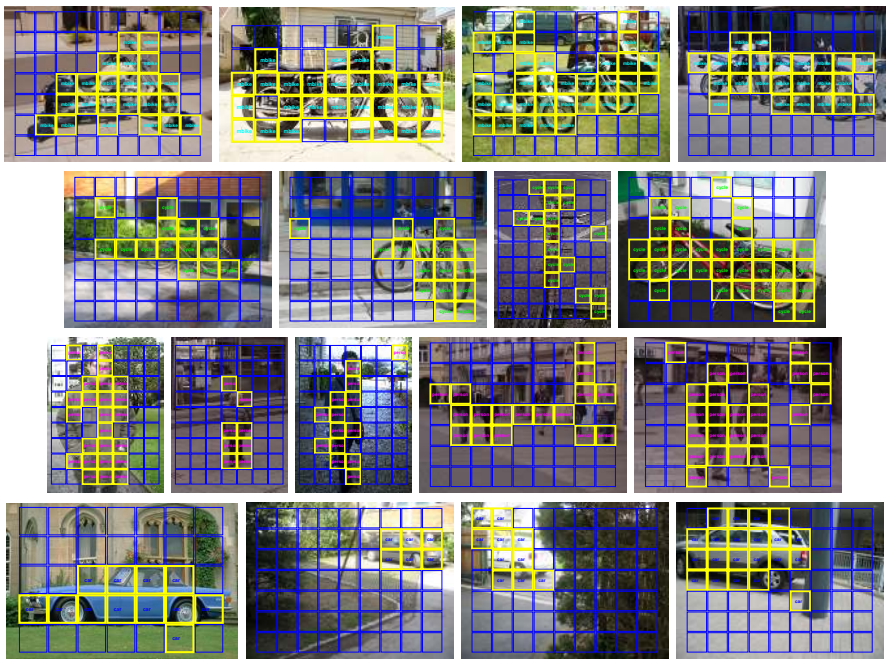


Fig. 7. Object localization in the PASCAL dataset [10] by classifying local image regions using hyperfeatures. Each row shows examples of results using one of the four independent classifiers, each being trained to classify foreground regions of its own class against the combined set of all other regions – 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. Each region is classified independently – there is no attempt to enforce spatial coherence.

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 demonstrate the representational power of hyperfeatures, not to build a complete framework for object recognition, so the experiments below classify regions individually without any attempt to include top-down or spatial contiguity information.

The experiments shown here use the bounding boxes provided with the PASCAL 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. Fig. 7 shows results obtained

[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.

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**

Fig. 8. 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 negative best scores counting 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).

by using these one-against-all classifiers individually on the test images. 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 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. Hyperfeatures again add discriminative power to the base level features, giving an average gain of 4–5% in classification performance. Figure 8 shows the key entries of the combined and the two-class confusion matrices, with negatives being further broken down into 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. 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.

Future work: The hyperfeature idea is applicable to a wide range of problems involving part-based representations. In this paper the hyperfeature codebooks have been trained bottom-up by unsupervised clustering, but more discriminative training methods should be a fruitful area for future investigation. For example image class labels could usefully be incorporated into the learning of latent topics. We also plan to investigate more general LDA 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. Such models should allow us to model contextual information at several different levels of support, which may be useful for object detection.

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 thankful to Andrew Zisserman and Jitendra Malik for providing feedback on this work. Our experiments make use of code derived from C. Bouman's *Cluster* [6] for fitting Gaussian mixtures, D. Blei's implementation of LDA [5], and T. Joachim's SVM-Light [21]. See [2] for extra details on this work.

References

1. Center for Research in Language, International Picture Naming Project. Available from http://crl.ucsd.edu/ aszekely/ipnp/index.html.
2. A. Agarwal and B. Triggs. Hyperfeatures – Multilevel Local Coding for Visual Recognition. Technical report, INRIA Rhône Alpes, 2005.
3. 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.
4. A. Berg and J. Malik. Geometric Blur for Template Matching. In *Int. Conf. Computer Vision & Pattern Recognition*, 2001.
5. D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
6. C. A. Bouman. Cluster: An unsupervised algorithm for modeling Gaussian mixtures. Available from http://www.ece.purdue.edu/~bouman, April 1997.
7. W. Buntine and A. Jakaulin. Discrete principal component analysis. Technical report, HIIT, 2005.
8. W. Buntine and S. Perttu. Is multinomial pca multi-faceted clustering or dimensionality reduction? *AI and Statistics*, 2003.
9. J. Canny. Gap: A factor model for discrete data. In *ACM Conference on Information Retrieval (SIGIR)*, Sheffield, U.K., 2004.
10. Visual Object Classes Challenge. The PASCAL Object Recognition Database Collection. Available at www.pascal-network.org/challenges/VOC.
11. G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *European Conf. Computer Vision*, 2004.
12. G. Dorko and C. Schmid. Object class recognition using discriminative local features. Technical report, INRIA Rhône Alpes, 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. R. Fergus and P. Perona. The Caltech database. Available at www.vision.caltech.edu/html-files/archive.html.
16. M. Fritz, E. Hayman, B. Caputo, and J.-O. Eklundh. The KTH-TIPS database. Available at www.nada.kth.se/cvap/databases/kth-tips.
17. 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.
18. 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.
19. C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Alvey Vision Conference*, pages 147–151, 1988.

20. T. Hofmann. Probabilistic Latent Semantic Analysis. In *Proc. of Uncertainty in Artificial Intelligence*, Stockholm, 1999.
21. T. Joachims. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.
22. F. Jurie and B. Triggs. Creating Efficient Codebooks for Visual Recognition. In *Int. Conf. Computer Vision*, 2005.
23. T. Kadir and M. Brady. Saliency, Scale and Image Description. *Int. J. Computer Vision*, 45(2):83–105, 2001.
24. M. Keller and S. Bengio. Theme-Topic Mixture Model for Document Representation. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining*, 2004.
25. 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.
26. S. Lazebnik, C. Schmid, and J. Ponce. Affine-Invariant Local Descriptors and Neighborhood Statistics for Texture Recognition. In *Int. Conf. Computer Vision*, 2003.
27. 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.
28. Y. LeCun, F.-J. Huang, and L. Bottou. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *CVPR*, 2004.
29. T. Leung and J. Malik. Recognizing Surfaces Using Three-Dimensional Textons. In *Int. Conf. Computer Vision*, 1999.
30. D. Lowe. Distinctive Image Features from Scale-invariant Keypoints. *Int. J. Computer Vision*, 60, 2:91–110, 2004.
31. J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *J. Optical Society of America*, A 7(5):923–932, May 1990.
32. K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 27(10), 2005.
33. 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.
34. G. Mori and J. Malik. Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. In *Int. Conf. Computer Vision & Pattern Recognition*, 2003.
35. A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. The Graz image databases. Available at http://www.emt.tugraz.at/∼pinz/data/.
36. 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.
37. J. Puzicha, T. Hofmann, and J. Buhmann. Histogram Clustering for Unsupervised Segmentation and Image Retrieval. *Pattern Recognition Letters*, 20:899–909, 1999.
38. M. Riesenhuber, T., and Poggio. Hierarchical Models of Object Recognition in Cortex. *Nature Neuroscience*, 2:1019–1025, 1999.
39. F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Int. Conf. Computer Vision*, pages 636–643, Vancouver, 2001.
40. B. Schiele and J. Crowley. Recognition without Correspondence using Multidimensional Receptive Field Histograms. *Int. J. Computer Vision*, 36(1):31–50, January 2000.
41. B. Schiele and A. Pentland. Probabilistic Object Recognition and Localization. In *Int. Conf. Computer Vision*, 1999.
42. C. Schmid. Weakly supervised learning of visual models and its application to content-based retrieval. *Int. J. Computer Vision*, 56(1):7–16, 2004.
43. C. Schmid and R. Mohr. Local Grayvalue Invariants for Image Retrieval. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 19(5):530–534, 1997.
44. M. Varma and A. Zisserman. Texture Classification: Are filter banks necessary? In *Int. Conf. Computer Vision & Pattern Recognition*, 2003.