Segmentation Driven Object Detection with Fisher Vectors

Ramazan Gokberk Cinbis, Jakob Verbeek, Cordelia Schmid

Segmentation Driven Object Detection with Fisher Vectors

Ramazan Gokberk Cinbis Jakob Verbeek Cordelia Schmid

LEAR, INRIA Grenoble - Rhône-Alpes, France Laboratoire Jean Kuntzmann

firstname.lastname@inria.fr

Abstract

We present an object detection system based on the Fisher vector (FV) image representation computed over SIFT and color descriptors. For computational and storage efficiency, we use a recent segmentation-based method to generate class-independent object detection hypotheses, in combination with data compression techniques. Our main contribution is a method to produce tentative object segmentation masks to suppress background clutter in the features. Re-weighting the local image features based on these masks is shown to improve object detection significantly. We also exploit contextual features in the form of a full-image FV descriptor, and an inter-category rescoring mechanism. Our experiments on the PASCAL VOC 2007 and 2010 datasets show that our detector improves over the current state-of-the-art detection results.

1. Introduction

Object detection is an important computer vision problem, where the goal is to report both the location in terms of a bounding box, and the category of objects in an image. Significant progress has been made over the past decade, as witnessed by the PASCAL VOC challenges [12]. Most of the existing work, see *e.g.* [10, 14], is based on the sliding window approach, where detection windows of various scales and aspect ratios are evaluated at many positions across the image. This approach becomes computationally very expensive when rich representations are used. To alleviate this problem, the seminal approach of Viola and Jones [37] implements a cascade, which iteratively reduces the number of windows to be examined. In a similar spirit, two or three-stage approaches have been explored [20, 35], where windows are discarded at each stage, while progressively using richer features. It is also possible to implement non-exhaustive search with a branch and bound scheme [24]. A recent alternative is to prune the set of candidate windows without using class specific information, by relying on low-level contours and image segmentation, see *e.g.* [1, 11, 17, 34]. In our work we use the method of [34].

Our first contribution is to explore the improved Fisher vector representation of [31] for object detection. This representation was recently shown to yield state-of-the-results for image and video categorization [4, 26]. Chen *et al*. [6] recently also explored Fisher vectors (FV) for detection, and proposed an efficient detection mechanism based on integral images to find the best scoring window per image. Their approach, however, does not allow the use of power and ℓ_2 normalization of the FVs. We show that this is a significant drawback, since these normalizations lead to substantially better detection performance when included.

Our second contribution is that we show that the image segmentation that drives the object hypotheses generation, can also be used to improve the appearance features computed over the windows. To this end, we compute a mask for each candidate window which counts for each pixel how many superpixels that cover that pixel are fully contained in the window, and weight the contribution of local descriptors in the Fisher vector representation accordingly. This local feature weighting process is class-independent, completely unsupervised, and suppresses background clutter on superpixels that traverse the window boundary.

Related work in the literature has used segmentation for object detection in different ways. Part of it, see *e.g.* [9, 27, 28, 38], extracts explicit segmentation for each object detection hypothesis as a post-processing step. Placing a bounding box over the obtained segmentation, however, can be sensitive to small defects in the segmentation. Moreover, if the supervision is limited to bounding box annotations, it is difficult to learn accurate object segmentation models. Other approaches, such as [18], instead rely on a bottom-up process which scores superpixels individually, and then assembles them into object detections. This approach has the drawback that the recognition of object fragments is much harder than recognizing complete objects. Recently, Fidler *et al*. [15] improve object detection using

the output from the semantic segmentation of [3]. The semantic segmentation is used to extract additional features encoding spatial relationships between the associated segments and object detection windows. This approach, however, requires groundtruth segmentations to train the semantic segmentation model.

Our work is different in the sense that we incorporate segmentation into the feature extraction step for object detection, and remain in the training-from-bounding-boxes paradigm. Even if the segmentation step fails in accurately delineating the object, our detector still benefits from the approximate segmentation since still part of the background clutter can be suppressed. Note that segmentation based post-processing may still be applied on top of our approach.

Our work is also related to recent work on weighting local features in representations for image classification. Khan *et al*. [23] use class-specific attention maps to weight local descriptors, and concatenate the class-specific bag-of-word histograms in their final representation. Sánchez *et al*. [30] sample 1,000 windows per image using the objectness measure of [1], and weight local features proportional to the number of windows that overlap them when computing a Fisher vector representation.

We evaluate our system using the PASCAL VOC 2007 and 2010 datasets, and compare it to results reported in the literature. To the best of our knowledge, our results are the best so far reported on these datasets, when considering the average performance across classes. With a gain of around 2 mAP points, our approximate segmentation masks significantly contribute to the success of our method.

In the next section we describe our method in detail, followed by the results of our experimental evaluation in Section 3. Finally, we present our conclusions in Section 4.

2. Segmentation driven object detection

In this section we describe how we generate our approximate segmentation masks, the feature extraction and compression processes, and the detector training procedure.

2.1. Segmentation mask generation

Hierarchical segmentation was proposed in [34] to generate class-independent candidate detection windows. The image is first partitioned into superpixels, which are then hierarchically grouped into a segmentation tree by merging neighboring and visually similar segments. This step is repeated using eight different sets of superpixels; obtained using four different color spaces and two different scale parameters for the superpixel generation. In this manner, a rich set of segments of varying sizes and shapes is obtained, and the bounding boxes of the segments are used as candidate detection windows. When producing around 1,500 object windows per image, more than 95% of the ground truth object windows are matched in the sense that they have an intersection/union measure of over 50%, as measured on the VOC'07 dataset. In this manner more computationally expensive classifiers and features can be used since far less windows need to be evaluated than in a sliding window approach. Examples of candidate windows together with their generating segments can be found in Figure 1.

In general, however, the segments used to generate these candidate windows do not provide good object segmentations. To obtain masks that are more suitable to improve object localization, we exploit the idea that background clutter is likely to be represented by superpixels that traverse the window boundary. Therefore, we produce a binary mask based on each of the eight segmentations by retaining the superpixels that lie completely inside the window, and suppressing the other ones. We average the eight binary masks to produce the weighted mask, which we use to weight the contribution of local features in the window descriptor. The procedure is illustrated in Figure 1. The segmentation quality varies across the eight segmentations from one image to another, but the average mask produces a relatively high quality segmentation for the correct object hypotheses shown in the first two rows, in particular considering that the method is completely unsupervised and class-independent.

It is important to consider the segmentation masks produced for incorrect candidate windows too, since these represent the vast majority of the candidate windows. For example, in the VOC 2007 dataset there are on average 2.5 objects per image, while we use on the order of 1,000 to 2,000 candidate windows per image. The first incorrect candidate window in Figure 1 shows a case where a partially visible horse is largely suppressed, since the superpixels on the object straddle outside the window. As a result this window gets a lower score than the correct one containing the entire horse. The second incorrect window shows a case where the car features are retained, and background is suppressed. Since the window does not accurately cover the object, this might be detrimental to the detector performance. It is, therefore, important to also take into account the features of the entire window as shown experimentally in Section 3.

2.2. Feature extraction

To represent the candidate object windows we use two local features: SIFT and the local color descriptor of [8]. Both descriptors are extracted on a dense multi-scale grid, with step size equal to 25% of the patch width, and on 16 scales separated by a factor 1.2, with 12×12 patches at the smallest scale. We project both features to $D = 64$ dimensions using PCA.

We aggregate the local feature vectors using the Fisher vector (FV) representation [31]. This representation extends the well-known bag-of-words representation by extracting first-order and second-order moments from the descriptors assigned to a visual word in addition to the number

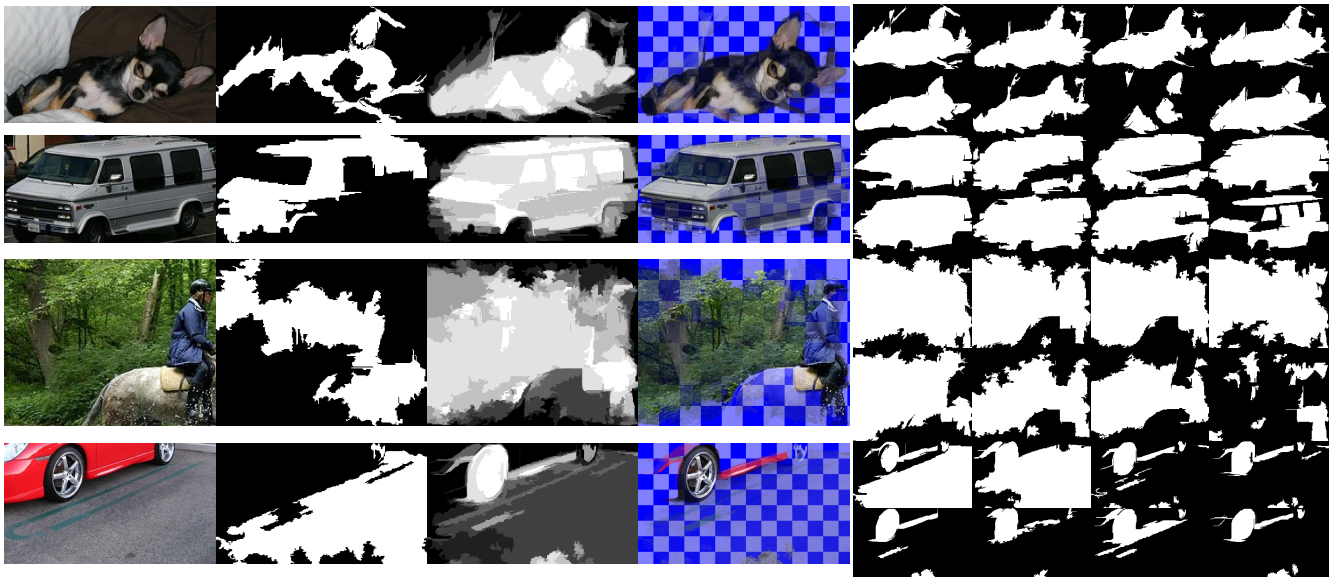

Figure 1: Segmentation masks for two correct (top) and two incorrect (bottom) candidate windows. The first four columns show the window, the merged segment that produced that window, our weighted mask, and the masked window. The eight images on the right show the binary masks of superpixels lying fully inside the window, for each of the eight segmentations.

of assigned descriptors. Formally, the FV is defined as the gradient of the log-likelihood of the data under a generative model, normalized by the inverse square-root of the Fisher information matrix. We learn a mixture of Gaussian (MoG) distribution $p(\boldsymbol{x})$ with diagonal covariance matrices from a large collection of 10^6 local descriptors off-line. The normalized gradients with respect to μ_{kd}, and σ_{kd} —the mean and standard deviation of the k-th Gaussian in dimension d— for a single local descriptor $\boldsymbol{x}$ are given by:

$$\frac{\partial \ln p(\boldsymbol{x})}{\partial \mu_{kd}} = \frac{p(k|\boldsymbol{x})}{\sqrt{\pi_k}} \left(\frac{x_d - \mu_{kd}}{\sigma_{kd}} \right), \quad (1)$$

$$\frac{\partial \ln p(\boldsymbol{x})}{\partial \sigma_{kd}} = \frac{p(k|\boldsymbol{x})}{\sqrt{\pi_k}} \left(\frac{(x_d - \mu_{kd})^2}{\sigma_{kd}^2} - 1 \right), \quad (2)$$

where π_k is the mixing weight of the k-th Gaussian, and $p(k|\boldsymbol{x})$ is the soft-assignment of $\boldsymbol{x}$ to the k-th Gaussian. In practice we use a hard-assignment instead of the posterior to speed-up feature calculation for many candidate windows.

To represent a candidate window we sum up these normalized gradients, and weight the contribution of local descriptors by the averaged segmentation masks when we use them. After aggregating the local gradient vectors, we obtain a $K(2D+1)$ dimensional feature vector to which we apply the power and ℓ_2 normalization [31]. The power normalization consists in taking a "signed square-root", $z \leftarrow \text{sign}(z)\sqrt{\text{abs}(z)}$, on each dimension separately. In [7], power normalization was shown to provide an approximation to the fisher vectors w.r.t. an improved MoG-like model incorporating dependencies across image patches.

To obtain the final window descriptor we concatenate the FVs obtained over the color and SIFT features. We also employ a form of rigid spatial layout [25], and compute FVs over cells in a 4×4 grid over the window; we also do this in combination with our masks. To capture global scene context, we compute a FV over the full image. In our experiments we assess their relative importance of these features.

2.3. Feature compression

During training we apply our detectors several times to the training images to retrieve hard negative examples. Re-extracting descriptors at each hard negative mining iteration would be very costly. For example, the PASCAL VOC 2007 dataset contains about 5,000 training images and we have between 1,000 and 2,000 candidate windows per image, thus we have to assess in the order of 5 to 10 million candidate windows in each iteration. On the other hand, storing all window descriptors in memory is also problematic. In our experiments we use $K = 64$ Gaussians, which leads to $K(2D+1) = 8,256$ dimensional FVs, which for 5 million candidate windows represents about 160 GB when using 4-byte floating point encoding. When using more elaborate descriptors, *e.g.* when including color, spatial pyramids, or masks, the memory usage becomes quickly prohibitive.

To overcome this problem, we compress the feature vectors using product quantization (PQ), which was recently proposed for large-scale image retrieval and classification [21, 29]. In product quantization, the large H dimensional feature vector is split into B subvectors, and a separate k-means quantizer with 2^M centers is learned for each subvector. A high dimensional vector can then be compressed

to $B \times M$ bits, by encoding for each subvector the index of the nearest k-means center. In practice we use $M = 8$, and $H/B = 8$ dimensional subvectors, which leads to a compression factor of 32 as compared to a 4-byte floating point encoding of the original vector. To reduce the memory requirements even further, we use Blosc compression [2] on the PQ codes per image. Blosc is a highly-optimized lossless data compression algorithm with low computational overhead, which exploits regularities across the descriptor PQ codes.[1] Note that PQ compression was used for object detection before in [36], but for a HOG feature based system which is far less demanding in terms of storage. We compare to their results in our experimental evaluation.

One can choose not to compress the data during test time, and apply the detector in an online manner computing the features for one image at the time. In our experimental setup, however, we have used the same compression approach on the test images; in [29] it was shown that this only has a small impact on performance. Using PQ codes, all window descriptors for the whole dataset take 580 GB of disk space. Blosc compression further reduces the data size roughly by a factor 4, down to 137 GB.

In order to apply a detector (for hard negative mining or evaluation), we only need to decompress Blosc-compressed data on-the-fly back to PQ codes. The PQ codes can be used directly to score windows efficiently using lookup tables [36]. Once data is loaded into memory, applying a detector on 5,000 images takes around 5 minutes using 35 cores for a single category and around 20 minutes for all 20 categories. Since most time is spent reading data from memory and decompressing the Blosc codes, the detection time scales sub-linearly with the number of categories.

2.4. Training the detector

For each category we train a linear SVM classifier on the concatenated FV representation of the windows. As positive training examples, we use the windows given by the ground-truth annotation. We initialize the set of negative training examples by randomly sampling candidate boxes around ground-truth windows, and retaining those windows that have an overlap between 20% and 30% with a positive example in terms of intersection over union.

After the initial training stage, we add hard negative examples by applying the detector on the training set. At each hard negative mining iteration, we select the top two detections per image, with less than 30% overlap with any ground-truth window. To avoid redundancy in negative samples, we do not allow two negative windows to have more than 60% overlap.

Using our development dataset, described in the next section, we observed that the detector performance significantly increases after the first hard negative mining iteration, and usually stabilizes after a few iterations. Based on this observation we fixed the number of hard negative mining iterations to four in all our experiments.

To learn the classifiers from the PQ compressed data, we use the dual coordinate descent algorithm of LibLinear [13] which updates the classifier after accessing a single example at a time. We modified the code to decompress examples on-the-fly as they are accessed by the training algorithm.

[1] We use the public code from http://blosc.pytables.org.

3. Experimental evaluation

We conduct experiments on the PASCAL VOC datasets of 2007 and 2010 [12]. To develop our approach and to evaluate the different variants of the approach, we use a subset of 1,000 images of the classes *bus*, *cat*, *motorbike*, and *sheep* from the "train+val" part of the 2007 dataset. These 1,000 images are again split into equal train and test sets; experiments on this development set do not use any images of the "test" set of the 2007 dataset.

For SVM training, there are two important hyper-parameters to set. The first one determines the balance between positive and negative examples, and the second one is the weight of the regularization term. On the development dataset, we have observed that using a fixed set of parameters performed as well as cross-validating these parameters per class. Therefore, in all experiments below, we have set the total weight of negative examples to be 100 times larger than the total weight of all positive examples, and set the weight of the SVM's ℓ_2 regularization term to 10^{-2}.

3.1. Parameter evaluation on the development set

We evaluated different versions of our detector on the development set, the results of which can be found in Table 1.

In our first three experiments we consider different detectors that only rely on the candidate windows, and do not make use of segmentation masks. We start with a basic detector that computes a single (power and ℓ_2 normalized) Fisher vector (FV) over the SIFT descriptors in each window , which leads to an mAP of 25.2%. When adding a 4×4 SPM grid, and concatenating the $1 + 4 \times 4 = 17$ FVs, the detection mAP value improves to 44.2%, underlining the importance to take spatial information into account. Next, we consider applying the ℓ_2 normalization per spatial cell instead of on the concatenated vector. We observe a small improvement to 45.0% mAP.

In order to evaluate the importance of descriptor normalization, we removed the power normalization and test three versions: (i) no normalization (*i.e.*, just summing the per-descriptor Fisher vectors), (ii) normalize (*i.e.* divide) by the number of local descriptors, (iii) using ℓ_2 normalization. This results in 3.4%, 6.9%, and 42.7% mAP, respectively (not shown in Table 1). Compared to the version with power and ℓ_2 normalizations, 45.0% mAP, it is clear than both normalization techniques are important for the detection task.

Table 1: Performance on the development set with different descriptors (S: SIFT, C: color), regions (W: window, G: generating segment, M: mask), and with / without SPM.

Desc.	Regions	Norm.	SPM	bus	cat	mbike	sheep	mAP
S	W	object	no	22.2	35.8	26.3	16.6	25.2
S	W	object	yes	47.6	45.0	54.2	30.0	44.2
S	W	cell	yes	48.0	47.2	53.0	32.0	45.0
S	G (train on W)	cell	yes	35.7	46.3	43.2	17.0	35.5
S	M (train on W)	cell	yes	41.1	47.8	52.7	19.2	40.2
S	M	cell	yes	44.0	48.8	51.4	30.8	43.8
S	W+M	cell	yes	48.5	49.2	54.3	33.8	46.4
S+C	W	cell	yes	47.3	48.2	54.4	35.8	46.4
S+C	W+M	cell	yes	48.1	51.1	**55.5**	40.0	48.7
S+C	W+M+F	cell	yes	**50.3**	**51.6**	54.8	**41.9**	**49.6**

The next four experiments in Table 1 assess the performance when using segmentation masks. First, we use for each window the generating segment used to produce it, *i.e*. the segments shown in the second column of Figure 1. In this case we suppress all descriptors within the bounding box that do not lie inside the segment, except for the ground-truth object windows during training, for which there are no generating segments. This leads to a detection mAP of 35.5%. Although this result is 10 mAP points below that using the window itself, it is still surprisingly good considering that the generating segments often poorly capture the object shape. Second, we repeat this experiment when using the weighted masks (see Figure 1 third column), which improves mAP by about five points to 40.2%. Third, we also use the weighted masks on the ground-truth object windows during training. This improves the detector to 43.8% mAP, due to a better match between training and test data. This is, however, slightly lower than the results obtained from the windows. This might be due to the fact that useful contextual background descriptors tend to be suppressed. Our last experiment in this set considers combining the mask and window descriptors, so as to benefit from both local context, and crisper object-centered features. This combination outperforms the window-only detector on all four classes and leads to 46.4% mAP.

In the last three experiments, we examine the added value of additional color and full-image features. For both of these we do not apply SPM grids. First, we consider adding color to both the window-only detector and the window+mask detector. The window-only SIFT+color detector performs very similar to the window+mask SIFT-only detector at 46.4% mAP. When adding color to the window+mask detector performance rises to 48.7%, clearly showing the complementarity of the mask and color features. Finally, we add a contextual feature by means of a FV computed over the full image, which further increases the mAP score to 49.6%.

3.2. Evaluation and comparison to existing work

We now turn to evaluations on the full PASCAL VOC 2007 and 2010 datasets. Based on the above experiments we use SPM, power and cell-level ℓ_2 normalization.

In Table 2 we present results obtained for various versions of our detector on the 2007 dataset. First, we consider the window-only version, which obtains 34.0% mAP. Second, we consider the combined window+mask version, which obtains an mAP of 35.8%. In the following two rows, we repeat the first two experiments with additional color features, which score 35.2% and 37.6% mAP, respectively. These relative performances are consistent with observations made on the development set. When we add the full-image FV to window+mask descriptors, mAP score is increased from 37.6% to 38.5%. To confirm the gain due to use of masks, we also report results for (SIFT+color, window+full), which is 36.6% mAP. Thus, the gain by adding masked features is consistently around 2 mAP points. Finally, we implement the contextual rescoring mechanism proposed in [14], which further increases the score from 38.5% to 40.5% mAP (last row).

To gain insight in the effect of the masked features, we present top detections in example images with our best detector (SIFT+color, window+full+context) with and without masks in Figure 2. Images in the top row illustrate cases where the detector benefits from the masked features. Our approximate object segmentation suppresses background clutter, which is particularly important when the object does not fill the bounding box (columns 1–4). The bus example shows a case where a too small detection is suppressed since superpixels extend over the full bus, using the mask leads to the full bus being detected. The bottom row shows examples where the use of masks degrades the top detection, typically the detection window is too large, since included background features are suppressed by the masks.

In Table 3 we compare our results to those of eight representative state-of-the-art detectors. We divide them in two groups depending on whether they exploit inter-class contextual features, which we refer shortly as *contextual detectors*, or score windows independently. We can observe that our detector without contextual rescoring obtains 38.5% mAP, which is significantly better than the highest mAP (34.8%) among the competing non-contextual detectors, and very similar to the highest mAP (38.7%) among the competing contextual detectors. With contextual rescoring, our detector obtains 40.5% mAP, which is the highest reported result on this dataset to the best of our knowledge.

Since we use the candidate window method of Van de Sande *et al*. [34], the detectors can be directly compared. In their work they used intersection-kernel SVM classifiers on bag-of-word representations with a 4-level spatial pyramid, computed over SIFT, and two color features (opponent-SIFT, and RGB-SIFT). Our detector without contextual

Table 2: Performance on VOC'07 with different descriptors (S: SIFT, C: color), regions (W: window, M: mask, F: full image).

		aero	bicy	bird	boa	bot	bus	car	cat	cha	cow	dtab	dog	hors	mbik	pers	plnt	she	sofa	trai	tv	mAP
S	W	46.7	48.7	14.1	19.4	15.7	45.0	54.6	36.3	11.4	36.2	37.4	24.3	37.1	52.4	25.8	14.7	35.3	30.4	47.2	48.2	34.0
S	W+M	50.2	49.4	16.6	21.3	15.7	45.5	55.3	39.8	14.8	36.3	39.5	25.4	42.4	50.4	**30.6**	15.8	34.3	35.5	48.3	49.7	35.8
S+C	W	47.7	50.1	16.5	19.2	15.9	45.1	55.1	37.2	13.0	37.3	40.8	25.5	40.7	51.8	26.4	18.2	35.5	30.6	47.7	49.6	35.2
S+C	W+M	50.5	51.2	18.8	23.8	17.8	47.2	56.4	41.6	14.7	38.6	40.7	27.1	47.3	52.4	29.7	19.6	38.3	35.0	49.3	52.8	37.6
S+C	W+F	49.9	51.6	16.4	21.7	16.5	45.9	55.6	38.4	15.3	**42.1**	42.0	25.3	41.2	52.2	26.8	18.8	36.2	35.8	48.5	51.6	36.6
S+C	W+M+F	52.6	52.6	19.2	25.4	18.7	47.3	56.9	42.1	**16.6**	41.4	41.9	27.7	47.9	51.5	29.9	20.0	**41.1**	36.4	48.6	53.2	38.5
S+C	W+M+F+Context	**56.1**	**56.4**	**21.8**	**26.8**	**19.9**	**49.5**	**57.9**	**46.2**	16.4	41.4	**47.1**	**29.2**	**51.3**	**53.6**	28.6	**20.3**	40.5	**39.6**	**53.5**	**54.3**	**40.5**

Table 3: Comparison of our detector with and without context with the state-of-the-art object detectors on VOC 2007.

	aero	bicy	bird	boa	bot	bus	car	cat	cha	cow	dtab	dog	hors	mbik	pers	plnt	she	sofa	trai	tv	mAP
	methods without inter-class contextual cues																				
SUGS'11 [34]	43.3	46.4	11.2	11.9	9.3	49.3	53.7	39.2	12.5	36.8	42.0	26.4	47.0	52.1	23.5	11.9	29.7	36.1	42.0	48.7	33.7
HMR'12 [19]	23.3	41.0	9.9	11.0	17.0	37.8	38.4	11.5	11.8	14.5	12.2	10.2	44.8	27.9	22.4	3.1	16.3	8.9	30.3	28.8	21.0
VZ'12 [36]	27.9	55.2	9.5	10.4	16.4	47.6	52.0	16.0	13.5	18.6	20.7	10.7	53.4	39.7	37.3	10.4	12.7	19.7	41.7	40.9	27.7
GFM'12 [16]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
KAWBVL'12 [22]	34.5	61.1	11.5	19.0	22.2	46.5	58.9	24.7	21.7	25.1	27.1	13.0	59.7	51.6	44.0	19.2	24.4	33.1	48.4	49.7	34.8
SWJZ'13 [32]	35.3	60.2	16.6	**29.5**	**53**	57.1	49.9	**48.5**	11	23	27.7	13.1	58.9	22.4	41.4	16	22.9	28.6	37.2	42.4	34.7
Ours, without context	52.6	52.6	19.2	25.4	18.7	47.3	56.9	42.1	16.6	**41.4**	41.9	27.7	47.9	51.5	29.9	20.0	**41.1**	36.4	48.6	53.2	38.5
	methods using inter-class contextual cues																				
GFM'12 context [16]	36.6	62.2	12.1	17.6	28.7	54.6	60.4	25.5	21.1	25.6	26.6	14.6	60.9	50.7	44.7	14.3	21.5	38.2	49.3	43.6	35.4
CDXH'13 [5]	41.0	**64.3**	15.1	19.5	33.0	**57.9**	**63.2**	27.8	23.2	28.2	29.1	16.9	**63.7**	**53.8**	**47.1**	18.3	28.1	**42.2**	53.1	49.3	38.7
Ours, with context	**56.1**	56.4	**21.8**	26.8	19.9	49.5	57.9	46.2	16.4	**41.4**	**47.1**	**29.2**	51.3	53.6	28.7	**20.3**	40.5	39.6	53.5	**54.3**	**40.5**

rescoring outperforms theirs on 17 of the 20 categories, as well as on average, 38.5% vs. 33.7% mAP.

Among the methods without context best results are obtained with [22], which introduces high-level color-name features in the deformable part-based model (DPM) of [14]. Our detector mAP compares favorably to theirs on average, 38.5% *vs*. 34.8% mAP, as well as on 13 of the 20 classes.

Chen *et al*. [5] propose a technique for exploiting contextual information from a single-category detector output. Although the method itself does not directly use inter-class information, they utilize the detections given by the contextual rescoring approach of [16]. Our non-contextual detector performs comparably at 38.5% mAP, and our contextual detector performs significantly better at 40.5% mAP compared to their 38.7%. In principle their method is generic, therefore, potentially additional gains may be achieved by applying their method on the top of our object detector.

Finally, in Table 4 we compare our performance on the PASCAL VOC 2010 dataset to earlier results, again dividing existing methods based on whether or not they use inter-class contextual features. For completeness, we also report the non-contextual and contextual detection results of [15], which are incomparable to the other results in the table, since their method is based on a semantic segmentation model trained using manual segmentation annotations for the training images of both the detection and the segmentation challenges of PASCAL VOC 2010.

On the PASCAL VOC 2010 dataset, we obtain an mAP score of 35.8% without contextual rescoring. Our non-contextual detector outperforms all other non-contextual detectors in terms of mAP, and performs comparable to other contextual object detectors. When contextual rescoring is used, detection performance of our method increases to 38.4% mAP, which to the best of our knowledge is 1.6% mAP points above the highest reported result (36.8% mAP) on the PASCAL VOC 2010 dataset.

Compared to the results of Van de Sande *et al*. [34], which are based on the same candidate windows, our non-contextual detection results are better than theirs on 12 of the 20 categories, as well as on average: 35.8% *vs*. 34.1%.

Compared to the two best previous methods, NLPR and those of Song *et al*. [33], which include inter-class context, our mAP score of 38.4% is significantly higher than their 36.8%. In a per-class comparison our system is best on 11 of the 20 categories, NLPR on 3, and Song *et al*. [33] on 4.

4. Conclusions

We presented an object detection approach that exploits the powerful high dimensional Fisher vector representation. We use a selective search strategy and data compression to efficiently train and test our detector. We have shown that the same superpixels that drive the selective search can

Table 4: Comparison of our detector with and without context with the state-of-the-art object detectors on VOC 2010.

	aero	bicy	bird	boa	bot	bus	car	cat	cha	cow	dtab	dog	hors	mbik	pers	plnt	she	sofa	trai	tv	mAP
	methods without inter-class contextual cues																				
SUGS'11 [34]	58.2	41.9	19.2	14.0	14.3	44.8	36.7	48.8	12.9	28.1	28.7	39.4	44.1	52.5	25.8	**14.1**	38.8	34.2	43.1	42.6	34.1
GFM'12 [16]	45.6	49.0	11.0	11.6	27.2	50.5	43.1	23.6	17.2	23.2	10.7	20.5	42.5	44.5	41.3	8.7	29.0	18.7	40.0	34.5	29.6
SWJZ'13 [32]	44.6	48.5	12.9	**26.3**	47.5	41.6	45.3	39	10.8	21.6	23.6	22.9	40.9	30.4	37.9	9.6	17.3	11.5	25.3	31.2	29.4
Ours, without context	61.3	46.4	21.1	21.0	18.1	49.3	45.0	46.9	12.8	29.2	26.1	38.9	40.4	53.1	31.9	13.3	39.9	33.4	43.0	45.3	35.8
	methods using inter-class contextual cues																				
NLPR 2010 *	53.3	**55.3**	19.2	21.0	30.0	54.4	46.7	41.2	**20.0**	31.5	20.7	30.3	48.6	55.3	**46.5**	10.2	34.4	26.5	50.3	40.3	36.8
SCHHY'11 [33]	53.1	52.7	18.1	13.5	**30.7**	53.9	43.5	40.3	17.7	31.9	28.0	29.5	**52.9**	**56.6**	44.2	12.6	36.2	28.7	**50.5**	40.7	36.8
GFM'12 context [16]	48.2	52.2	14.8	13.8	28.7	53.2	44.9	26.0	18.4	24.4	13.7	23.1	45.8	50.5	43.7	9.8	31.1	21.5	44.4	35.7	32.2
Ours, with context	**65.9**	50.1	**23.7**	24.1	20.4	52.6	**47.1**	**50.9**	13.2	**32.8**	**31.8**	**41.4**	43.9	55.3	29.8	**14.1**	**41.7**	**35.6**	46.7	**46.9**	**38.4**
	uncomparable methods using additional training data																				
FMYU'13 [15] **	56.4	48.0	24.3	21.8	31.3	51.3	47.3	48.2	16.1	29.4	19.0	37.5	44.1	51.5	44.4	12.6	32.1	28.8	48.9	39.1	36.6
FMYU'13 context [15] **	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4

* These results do not directly correspond to papers and are taken directly from the PASCAL VOC 2010 website instead.
** Utilizes groundtruth segmentation annotations and extra training images.

be used to obtain approximate object segmentation masks, which allow us to compute object-centric features that are complementary to full-window features. Our detector also exploits contextual features in the form of a full-image FV descriptor, and an inter-category rescoring mechanism. [2]

To the best of our knowledge, our detection results on the PASCAL VOC 2007 and 2010 datasets are the best so far reported on these datasets, when considering the average performance across classes. With a gain of around 2 mAP points, our approximate segmentation masks significantly contribute to the success of our method.

In future work we want to explore the effectiveness of our approximate object detection masks for tasks such as semantic segmentation, by using them as a strongly semantic and spatially detailed prior.

Acknowledgements. This work was supported by Quaero (funded by OSEO, French State agency for innovation), the European integrated project AXES, and the ERC advanced grant ALLEGRO.

References

[1] B. Alexe, T. Deselares, and V. Ferrari. Measuring the objectness of image windows. *PAMI*, 34(11):2189–2202, 2012.

[2] F. Alted. Why modern CPUs are starving and what can be done about it. *Computing in Science & Engineering*, 12(2):68–71, 2010.

[3] J. Carreira, R. Caseiroa, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012.

[4] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[2] We are indebted to M. Douze for helping to release the code publicly available at: http://lear.inrialpes.fr/src/maskfishdet.

[5] G. Chen, Y. Ding, J. Xiao, and T. X. Han. Detection evolution with multi-order contextual co-occurrence. In *CVPR*, 2013.

[6] Q. Chen, Z. Song, R. Feris, A. Datta, L. Cao, Z. Huang, and S. Yan. Efcient maximum appearance search for large-scale object detection. In *CVPR*, 2013.

[7] R. Cinbis, J. Verbeek, and C. Schmid. Image categorization using Fisher kernels of non-iid image models. In *CVPR*, 2012.

[8] S. Clinchant, G. Csurka, F. Perronnin, and J.-M. Renders. XRCE's participation to ImagEval. In *ImageEval workshop at CVIR*, 2007.

[9] Q. Dai and D. Hoiem. Learning to localize detected objects. In *CVPR*, 2012.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[11] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*, 2010.

[12] M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) challenge. *IJCV*, 88(2):303–338, June 2010.

[13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: a library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[14] P. Felzenszwalb, R. Grishick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *PAMI*, 32(9), 2010.

[15] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun. Bottom-up segmentation for top-down detection. In *CVPR*, 2013.

[16] R. Girshick, P. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5, 2012.

[17] C. Gu, P. Arbeláez, Y. Lin, K. Yu, and Malik. Multi-component models for object detection. In *ECCV*, 2012.

[18] C. Gu, J. Lim, P. Arbeláez, and J. Malik. Recognition using regions. In *CVPR*, 2009.

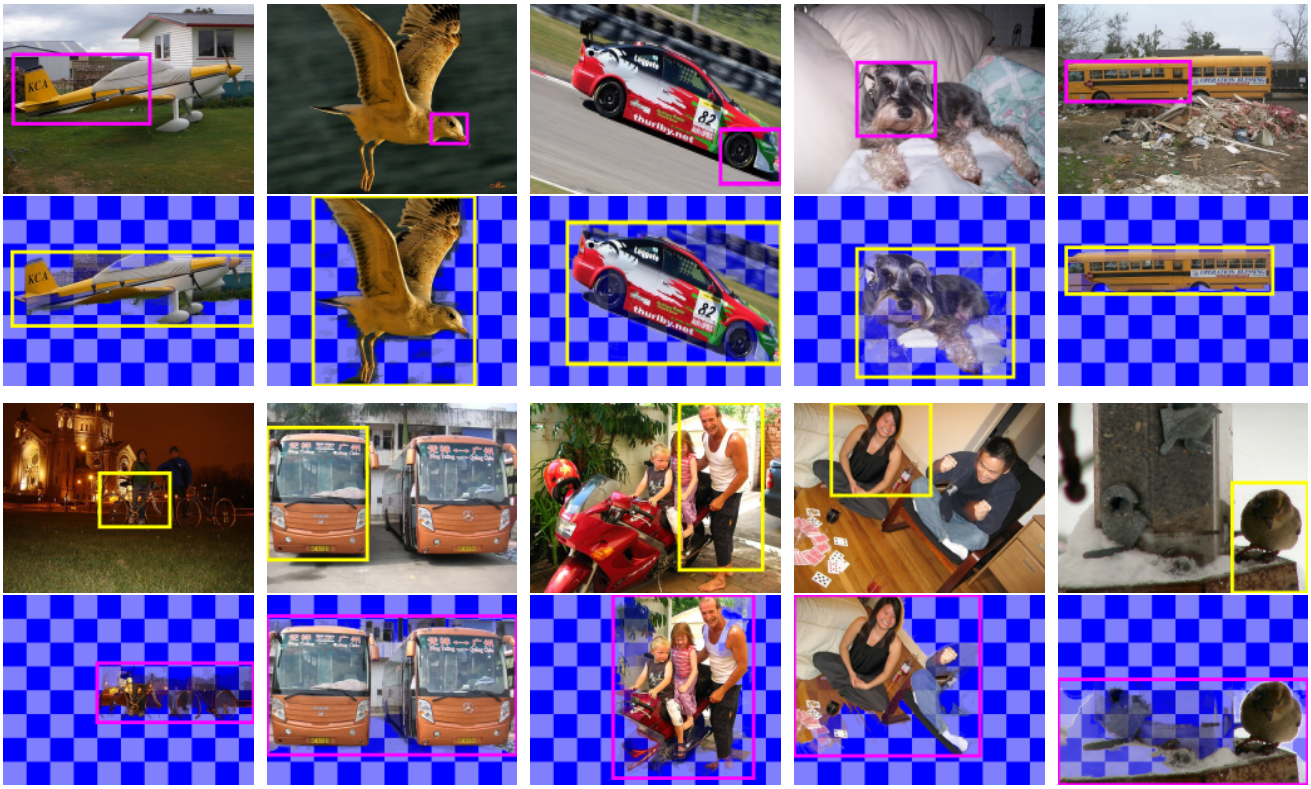

Figure 2: Example images where the top scoring detection improves (top row) or degrades (bottom row) with inclusion of the masked window descriptors. Correct detections are shown in yellow, incorrect ones in magenta. See text for details.

[19] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012.

[20] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.

[21] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *PAMI*, 33(1):117–128, 2011.

[22] F. Khan, R. Anwer, J. van de Weijer, A. Bagdanov, M. Vanrell, and A. Lopez. Color attributes for object detection. In *CVPR*, 2012.

[23] F. Khan, J. van de Weijer, and M. Vanrell. Top-down color attention for object recognition. In *ICCV*, 2009.

[24] C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: a branch and bound framework for object localization. *PAMI*, 31(12):2129–2142, 2009.

[25] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[26] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with Fisher vectors on a compact feature set. In *ICCV*, 2013.

[27] O. Parkhi, A. Vedaldi, C. Jawahar, and A. Zisserman. The truth about cats and dogs. In *ICCV*, 2011.

[28] D. Ramanan. Using segmentation to verify object hypotheses. In *CVPR*, 2007.

[29] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

[30] J. Sánchez, F. Perronnin, and T. de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33(16):2216–2223, 2012.

[31] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013.

[32] X. Song, T. Wu, Y. Jia, and S.-C. Zhu. Discriminatively trained and-or tree models for object detection. In *CVPR*, 2013.

[33] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan. Contextualizing object detection and classification. In *CVPR*, 2011.

[34] K. van de Sande, J. Uijlings, T. Gevers, and A. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011.

[35] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.

[36] A. Vedaldi and A. Zisserman. Sparse kernel approximations for efficient classification and detection. In *CVPR*, 2012.

[37] P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 57(2):137–154, 2004.

[38] L. Wang, J. Shi, G. Song, and I.-F. Shen. Object detection combining recognition and segmentation. In *ACCV*, 2007.