Fisher and VLAD with FLAIR

Koen E. A. van de Sande[1] Cees G. M. Snoek[1] Arnold W. M. Smeulders[12]
[1] ISLA, Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands
[2] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

Abstract

A major computational bottleneck in many current algorithms is the evaluation of arbitrary boxes. Dense local analysis and powerful bag-of-word encodings, such as Fisher vectors and VLAD, lead to improved accuracy at the expense of increased computation time. Where a simplification in the representation is tempting, we exploit novel representations while maintaining accuracy. We start from state-of-the-art, fast selective search, but our method will apply to any initial box-partitioning. By representing the picture as sparse integral images, one per codeword, we achieve a Fast Local Area Independent Representation. FLAIR allows for very fast evaluation of any box encoding and still enables spatial pooling. In FLAIR we achieve exact VLADs difference coding, even with ℓ_2 and power-norms. Finally, by multiple codeword assignments, we achieve exact and approximate Fisher vectors with FLAIR. The results are a 18x speedup, which enables us to set a new state-of-the-art on the challenging 2010 PASCAL VOC objects and the fine-grained categorization of the CUB-2011 200 bird species. Plus, we rank number one in the official ImageNet 2013 detection challenge.

1. Introduction

For object detection, action recognition, fine-grained image categorization and many other current topics, the trend is towards evaluating many candidate boxes in the image for the best result. This paper introduces a data structure, FLAIR, for which it is as efficient to evaluate one box as it is many boxes.

In spatial pyramids for image categorization the preferred number of boxes is 30 [24], due to the large computational load reduced to 17 in [5]. In the selective search algorithm for state-of-the-art object detection [24], the number of boxes is 2,000 per image. Ideally, including the 30 fine spatial pyramids leading to 60,000 boxes per image. Inevitably, computation has become the critical factor. Rather than re-computing the same facts many times, once per box, we reconsider the data structure of the image in order to compute evaluations once per image by a new representation of the data, FLAIR, the Fast Local Area Independent Representation (Figure 1).

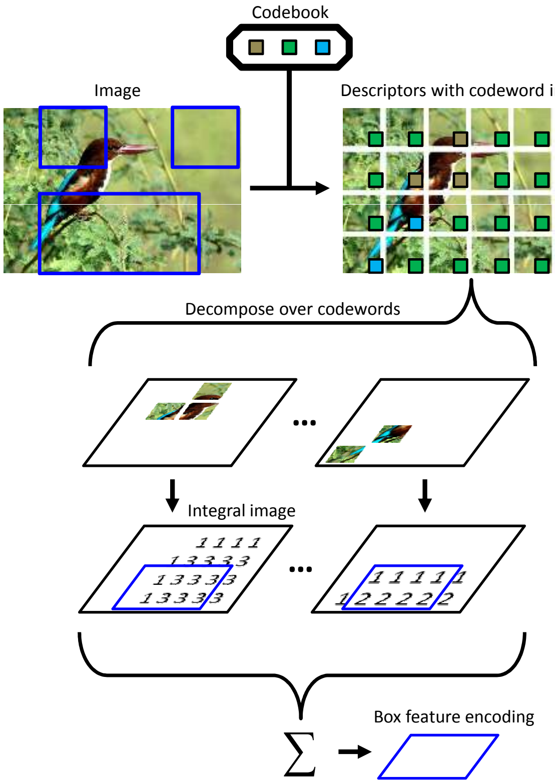

Figure 1. **Fast Local Area Independent Representation.** Given an initial box-partitioning, we represent a picture as sparse integral images, one per codeword dimension. FLAIR allows for very fast evaluation of any box encoding and still enables spatial pooling. In FLAIR we achieve exact VLADs difference coding, even with ℓ_2 and power-norms, as well as exact and approximate Fisher vectors.

In object detection, HOG has proven to be successful in combination with the part-based model by Felzenszwalb *et al*. [10]. It models object shape templates and scans the image with boxes at multiple scales. Because more than 100,000 boxes need to be inspected per object type and aspect ratio, the analysis must be restricted to the low-dimensional HOG features or to simple histograms. Recently, Dean *et al*. [8] report an impressive speed-up for object detectors based on HOG part-templates, but they still require exhaustive scanning. Simple histograms can be efficiently computed with multi-dimensional integral images [19], but use prohibitive amounts of memory at higher dimensionalities. Sub-window search [15] and selective search [24] opened the door to the use of locality with BoW, which is computationally more expensive than HOG [8] but superior in the quality of the semantic interpretation [15, 27, 12, 24, 25, 5]. Dean *et al.* achieve a speedup factor 20,000 at the cost of a drop in accuracy, we do a speedup factor of 18 but enabling an accuracy increase for the state-of-the-art in object detection.

The encoding of the appearance of points in the image to words has evolved from hard [7] and soft [26] coding to one cluster center, to VLAD [14], super vectors [33], and Fisher vectors [18, 22], which encode the difference between the point descriptors and the nearest cluster center(s). The last three representations have outperformed straight encodings. Difference codings impose an even heavier computational demand, as it requires a dimension-by-dimension comparison of the points in the bounding box. In FLAIR, we design the representation of the point appearances such that the Fisher vector and VLAD encoding becomes equal in speed with the straight BoW encoding, and gain their superior performance at no extra computational cost.

The advantage of the localized encoding is evident in fine-grained object detection [32], where Fisher with FLAIR greatly propels the likelihood of finding proper object correspondences between small, but often similar details, amidst the many distracting other details.

We start from the state-of-the-art selective search algorithm for object detection [24], but our method will apply to any initial box-segmentation of the image [1]. Our first novelty is an integral-image, area-independent representation, allowing for the fast evaluation of any set of boxes, including box candidate evaluation, and spatial pyramid pooling [16] (Section 3). Then, as the second novelty, we embed VLADs difference coding, ℓ_2 and power-norm into FLAIR, by introducing a multi-dimensional integral image per code word (Section 4). As third novelty we include descriptor assignment to multiple code words, enabling FLAIR also to Fisher vectors and other multiple word assignments (Section 5). The results are a gain of a factor 18 in algorithmic speed with the same accuracy. The speedup enables Fisher vectors with fine spatial pyramids for object detection, setting a new state-of-the-art on the challenging PASCAL VOC 2010 [9] as well as the CUB-2011 detection task of two hundred bird species [30] and it won the ImageNet 2013 detection challenge (Section 6).

2. Related Work

In many current algorithms in computer vision, the evaluation of many boxes in one image is the key to a high-level semantic interpretation of the image. Fidler *et al.* [11] start from supervised second order pooling [3]. The method is promising as the evaluation of the boxes leads to excellent detection results on PASCAL VOC 2010, (be it that it requires an additional round of annotations over the PASCAL annotation). We aim at a substantial improvement in the time it takes to evaluate boxes, to be able to evaluate more sophisticated features.

Faster ways to partition the image in overlapping boxes have been described in [1, 20], but their gain in speed comes with a loss in quality. Recently, Uijlings *et al.* [24] propose *selective search*, which combines multiple hierarchical divisions and five different criteria to arrive at a good set of candidate boxes. Selective search is fast, assures high-recall, and leads to an overlap with the object comparable with [10], but the algorithm is considerably more efficient than the reference. Selective search results in 2,000 boxes per image where [10] has 100,352. Moreover, selective search finds the boxes without any prior knowledge of the object type it searches for as in [11]. We select selective search as the method of box selection, but any other box-selection method can be applied in FLAIR as well.

Van de Sande *et al.* [25] further improves selective search by adding VLAD [14]. Recently, Cinbis *et al.* [5] achieves state-of-the-art detection on PASCAL VOC 2010 using selective search in combination with reweighted Fisher vectors [18]. In [5], the main computational bottleneck for detecting objects is the expensive encoding step for each box in the image. Moreover, in the references, Fisher and VLAD are shown to benefit from ℓ_2 or power-normalization, which implies that the feature vector of a box can no longer be merged from two smaller boxes. Hence in [25, 5] a brute-force approach is applied, made more efficient by the application of product quantization [13]. In contrast, FLAIR allows for fast encoding for VLAD or Fisher vectors on arbitrary boxes, while leaving the possibility intact to benefit from the recent advances in normalization [18, 14] and the advantages offered by fine spatial pooling [25, 21].

FLAIR enables the efficient evaluation of Fisher in many boxes and hence can be applied on the challenging problem posed in [30], where an uncropped bird image must be assigned to one of 200 bird species. Not only do we achieve automatic localization for this task, but in addition also a considerable improvement of the accuracy.

FLAIR rests on integral images and decomposition. Integral images allow for computing any sum of a rectangular area in constant time [29]. Apart from fast descriptor computation as applied in [20, 11], we note that the integral image is naturally suited for efficiently summing codeword counts inside a bounding box. In [19] integral images are extended to multiple dimensions so they efficiently construct histograms over rectangular regions. However, naively applying multi-dimensional integral images to VLAD and Fisher leads to prohibitive memory usage (>14GB) and datastructure creation time (minutes per image). In [15, 28], integral images have been used by a decomposition only suited for BoW and unnormalized features. The recent Fisher vector decomposition of Chen *et al*. [4] into its point-wise contributions is achieved by representing each point as a sparse vector of codeword-indices. The image is seen as a point-wise confidence map in which the most likely object location is searched for later [17]. Neither the integral representation of [15, 28], nor the point-wise representation in [4] allow for ℓ_2- and power-norms, and they do also not allow for the use of fine spatial pyramids. In all cases, this results in a considerable loss of accuracy compared to using them [18]. FLAIR achieves a large improvement in computation times while maintaining the state-of-the-art accuracy for local box-driven and fine spatial pyramids using VLAD and Fisher normalized vectors.

3. BoW with FLAIR

FLAIR enables fast construction of feature vectors for arbitrary boxes in the image. Given an image partitioning, which provides B bounding boxes, we first extract N descriptors in the full image. Since the state-of-the-art relies on dense sampling of descriptors [25, 5], N is proportional to the number of pixels in the image. The descriptor has dimensionality D. We first discuss box encoding with BoW.

Box Encoding with BoW BoW assigns the descriptor at location $\vec{v}_n$ to the closest word $\vec{c}_k$ in the codebook with size K within the Euclidian space:

$$\phi(\vec{v}_n, \vec{c}_k) = \begin{cases} 1 & \text{if } k \text{ is the closest codeword for } \vec{v}_n \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The cost of computing the Euclidian distances between K codewords and N descriptors (size D) is $O(KND)$.

The K-dimensional feature vector of BoW stores the number of descriptors hard assigned to each codeword k:

$$F_k = \sum_{n=1}^{N} \phi(\vec{v}_n, \vec{c}_k). \tag{2}$$

By looping over the N descriptors and counting for all codewords simultaneously in a single loop, this results in a complexity $O(N)$. The extension of Equation 2 to boxes is done by adding a membership test for the descriptor coordinates against the box coordinates. An algorithm for feature vector construction will loop through all descriptors for each box. Hence, for B boxes in an image, with a spatial pyramid per box consisting of S cells, it will have a complexity $O(NBS)$. This means that constructing the feature vector for a box depends on the size of the full image, which imposes a serious bottleneck.

To improve the efficiency, one could consider to first sort the feature vectors on their (y, x) coordinates. Then, only the rows within the box have to be considered. The size of the feature vector grows with the area of the box. In worst case, the box equals the image. Therefore, presorting does not change the upper bound of the complexity: $O(NBS)$.

Fast Local Area Independent Representation The key insight for FLAIR is that assignment of a descriptor to one of the codewords affects only a specific part of the feature vector. For BoW with hard assignment, each descriptor affects only one codebook element. Therefore, if we are interested in constructing the part of the feature vector corresponding to one codeword k, we only need to consider the descriptors for which k is the closest codeword. This allows us to decompose the problem of constructing the full feature vector into K smaller subproblems, where K is the size of the codebook. For each subproblem we compute Equation 2 for a single k. Concatenating the solutions to these subproblems will give us the full feature vector. From an algorithmic point of view, the decomposition over the words of the codebook is optimal in terms of the number of visits to the data [6].

An integral image restricts the number of visits to the data to two, sufficient to find the sum over a row. When an algorithm relies on evaluating the sum over (many) boxes, the complexity can be made independent of the area of the box, as illustrated in Figure 2. The value at coordinate (x, y) in the integral image is the sum of all the items above and to the left of (x, y):

$$I(x, y) = \sum_{x' < x, y' < y} i(x, y), \tag{3}$$

with $i(x, y)$ the number of descriptors at (x, y) that is closest to codeword k. Construction of the integral image is done in a single pass over the entire image using the fact that the value at (x, y) is simply:

$$I(x, y) = i(x, y) + I(x-1, y) + I(x, y-1) - I(x-1, y-1). \tag{4}$$

Let W be the width of the image and H the height. Then, the complexity of this pass is $O(WH)$, which is approximately $O(N)$, proportional to the number of pixels. As illustrated in Figure 2, evaluating any rectangle (x_1, y_1, x_2, y_2) requires just 4 operations with an integral image:

$$I(x_2, y_2) - I(x_1, y_2) - I(x_2, y_1) + I(x_1, y_1). \tag{5}$$

With integral images the complexity of evaluating a box is reduced to $O(1)$.

We have decomposed the problem of feature vector creation into K subproblems, one for each codeword. For every codeword we need to construct an integral image. These two components together form FLAIR: Fast, Local, Area Independent Representation. After codeword assignment, the cost of constructing FLAIR for BoW is $O(NK)$. FLAIR allows for creating feature vectors independent of the bounding box area: in each of the K integral images,

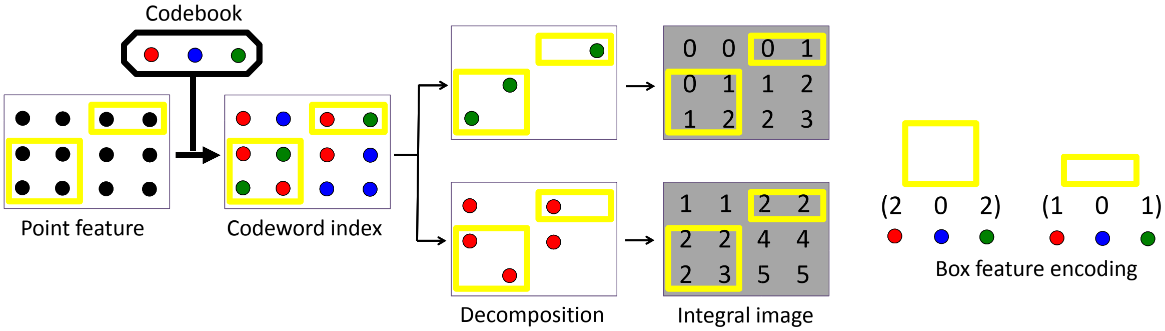

Figure 2. **BoW with FLAIR.** The key insight for FLAIR is that assignment of a descriptor to one of the codewords affects only a specific part of the feature vector. For BoW with hard assignment, each descriptor affects only one codebook element. This allows us to decompose the problem of constructing the full feature vector over codebook elements. For each individual codeword we construct an integral image, which counts the number of assigned points inside a bounding box. Computing the number of assigned points in an arbitrary box is just four operations (Equation 5). Concatenating the solutions to these subproblems will give us the full feature vector, with or without normalization (not shown). If one of the rows or columns in the decomposition is empty (*e.g.*, column three of the green codeword), then its values are simply a copy of the row or column to the left or above. For BoW, FLAIR will not help that much in practice. In this case, the gain of fast evaluation does not outweigh the fixed overhead of constructing the integral images. However, for more complex evaluations, as required for VLAD and Fisher vectors, FLAIR is advantageous.

we need just constant time. Evaluating all boxes and spatial pyramid cells with FLAIR has complexity $O(KBS)$. ℓ_p normalization of the feature vectors is simply included in FLAIR: sum the feature vector of a box and divide all elements by this sum. In summary, the computational complexity for creating BoW feature vectors has been lowered from $O(NBS)$ to $O(NK + KBS)$. With FLAIR, we have created a representation, which makes evaluating a box independent of its image area, while still allowing for normalization.

For BoW, FLAIR will not help that much in practice. In this case, the evaluation per box is relatively simple and the gain does not outweigh the fixed overhead of constructing the integral images. However, for more complex evaluations per pixels as required for VLAD and Fisher Vectors, FLAIR is advantageous, as we will discuss next.

4. VLAD with FLAIR

Similar to BoW, in VLAD descriptors are assigned to the closest codeword. In contrast to BoW, however, VLAD encodes the difference between the descriptor and the codeword. Hence, the descriptor $\vec{v}$ will affect D elements in the feature vector corresponding to the codeword k:

$$\vec{F}_k = \sum_{n=1}^{N} (\vec{v}_n - \vec{c}_k)\phi(\vec{v}_n, \vec{c}_k). \tag{6}$$

Similar to traditional BoW, we decompose the VLAD feature vector into K subproblems. For constructing the feature vector corresponding to codeword k, we need the descriptors for which k is the closest codeword. We use the same scheme, FLAIR, as discussed above, now extending the scalar integral image to multi-dimensional integral images, with x_n and y_n the coordinates of the n-th descriptor:

$$\vec{I}(x,y) = \sum_{n=1,\ \text{subject to}\ x_n<x,\ y_n<y}^{N} (\vec{v}_n - \vec{c}_k)\phi(\vec{v}_n, \vec{c}_k), \tag{7}$$

The extension of Equation 4 and 5 is trivial. The extension gives us VLAD with FLAIR.

Complexity analysis The computational complexity to constructing a multi-dimensional integral image is $O(KWHD)$ or $O(KND)$, where K is the number of codewords. N is proportional to the number of pixels and D is newly added to hold the dimension of the descriptor. Evaluating any rectangle requires four operations on vectors with D elements, *i.e.*, $O(D)$. The total cost of evaluating all boxes and pyramid cells in VLAD with FLAIR is $O(KDBS)$. This compares favorably with the ordinary evaluation of VLAD as we will discuss below.

Power and ℓ_p normalization Power and ℓ_2 normalization [14] are important for good recognition accuracy with VLAD. Power normalization combines easily with FLAIR, as the computation requires a read-out of all dimensions of the feature vector one by one, readily provided by the difference integral image. It is applied after a box is evaluated with the multi-dimensional integral image, *i.e.*, it does not depend on the solutions to other subproblems k. To also include ℓ_2 normalization in FLAIR requires more effort. As it is an important extension, we have nevertheless devised an extension to FLAIR to achieve that. As before, the computation of the ℓ_2 norm is decomposable into K subproblems. For a given part of the feature vector $\vec{F}_k$, the contribution to the ℓ_2 norm will be $(\vec{F}_k)^T\vec{F}_k$. Summing over all k and taking the square root gives the ℓ_2 norm of the full feature vector. Extensions to other ℓ_p norms are trivial. The complexity to create part of the ℓ_p norm is $O(D)$, the same as evaluating a box in the multi-dimensional integral image. Therefore, including ℓ_p and power-normalization of the feature vector in VLAD with FLAIR yields the same complexity.

Exploiting sparsity for memory efficiency The memory usage of difference integral images is $\theta(WHD)$ compared to $\theta(WH)$ for their scalar version. With an image of 500x375 pixels, $D = 80$, and $K = 256$, these datastructures use 57MB of memory for each k, equal to 14.3GB per image. Multi-dimensional integral images [19] have the same prohibitive memory usage. However, the memory requirements for FLAIR can be reduced significantly by skipping void rows and columns. When an image row or column contains no points to be included in the integral image, the values above or to the left in the integral image can be reused, see the example in Figure 2. If we let the number of rows with at least one point be $\hat{H}$ and the number of columns with at least one point to be $\hat{W}$, then a tighter bound on the computational complexity for constructing a multi-dimensional integral image is $O(\hat{W}\hat{H}D)$. In addition, by reusing void rows and columns through pointers, the memory per integral image decreases from $\theta(WHD)$ to $\theta(\hat{W}\hat{H}D)$. On average, 79% of rows and columns is void and memory usage drops to 1.0GB per image. With FLAIR, VLAD can be evaluated independent of the area of the boxes, requiring $O(K\hat{W}\hat{H}D + KDBS)$ time and $\theta(K\hat{W}\hat{H}D)$ memory.

5. Fisher with FLAIR

Where VLAD encodes just the descriptor differences, *i.e.*, the first order moments of a descriptor assigned to a codeword, the Fisher vector encoding includes first order and second order moments. The Fisher encoding is the normalized gradient of the log-likelihood of the data under a mixture of Gaussians distribution $p(\vec{v})$ with diagonal covariance matrices. The gradients for the k-th codeword, represented by a Gaussian with mean $\vec{\mu_k}$, standard deviations $\vec{\sigma_k}$ and mixing weight π_k against all descriptors are given by:

$$\frac{\Delta \ln p}{\Delta \vec{\mu}_k} = \frac{1}{N} \sum_{n=1}^{N} \frac{p(k|\vec{v}_n)}{\sqrt{\pi_k}} \left(\frac{\vec{v}_n - \vec{\mu}_k}{\vec{\sigma}_k} \right), \tag{8}$$

$$\frac{\Delta \ln p}{\Delta \vec{\sigma}_k} = \frac{1}{N} \sum_{n=1}^{N} \frac{p(k|\vec{v}_n)}{\sqrt{\pi_k}} \left(\frac{(\vec{v}_n - \vec{\mu}_k)^2}{\vec{\sigma}_k^2} - 1 \right), \tag{9}$$

where we abuse notation by defining all operations on vectors to be element-wise. The Fisher encoding assigns a descriptor to multiple codewords. However, because the assignment of a descriptor to one of the codewords k affects only specific parts of the feature vector, the decomposition into subproblems as done for VLAD is still possible. Consider:

$$S_0(k) = \sum_{n=1}^{N} p(k|\vec{v}_n), \tag{10}$$

$$\vec{S}_1(k) = \sum_{n=1}^{N} p(k|\vec{v}_n)\vec{v}_n, \tag{11}$$

(12)

$$\vec{S}_2(k) = \sum_{n=1}^{N} p(k|\vec{v}_n)\vec{v}_n^2. \tag{13}$$

With these functions we rewrite Equations 8 and 9:

$$\frac{\Delta \ln p}{\Delta \vec{\mu}_k} = \frac{1}{N\sqrt{\pi_k}\vec{\sigma}_k} \left(\vec{S}_1(k) - \vec{\mu}_k \cdot S_0(k) \right), \tag{14}$$

$$\frac{\Delta \ln p}{\Delta \vec{\sigma}_k} = \frac{1}{N\sqrt{\pi_k}} \left(\frac{\vec{S}_2(k) - 2\vec{\mu}_k \vec{S}_1(k) + \vec{\mu}_k^2 \cdot S_0(k)}{\vec{\sigma}_k^2} - S_0(k) \right). \tag{15}$$

The scalar integral image for $S_0(k)$ and the multi-dimensional integral images for $S_1(k)$ and $S_2(k)$ are supplemented by a scalar integral image holding the number of descriptors N in an area. With these four integral images the gradients for a single codeword evaluate in $O(D)$ and independent of the box area, similar to VLAD.

Complexity analysis The complexity to construct Fisher with FLAIR, *i.e.*, K times the four integral images is $O(KND)$. As with VLAD, Fisher with FLAIR creates feature vectors for B boxes with S cells in $O(KDBS)$. Power and ℓ_p normalization are included the same way as for VLAD with FLAIR. The computational complexity for Fisher has changed from $O(NDBS)$ in standard Fisher to $O(KND + KDBS)$ with FLAIR. With sparsity exploitation, as in VLAD, the order goes to $O(K\hat{W}\hat{H}D{+}KDBS)$.

Approximate Fisher (with FLAIR) In Fisher with FLAIR each descriptor is included in all of the K subproblems. Practical implementations of the Fisher encoding include a descriptor $\vec{v}$ for codeword k only if the posterior $p(\vec{v}|k)$ is higher than a threshold (typically 10^{-4}). Empirically, we find that a descriptor is assigned to on average 9.5 codewords for this threshold if $K = 256$. We can control the level of sparsity if we put a maximum T on the number of codewords to assign to. Up to this point all our solutions are exact, but this assignment limit introduces our first approximation: Approximate Fisher with FLAIR. In our experiments we will evaluate different T in terms of speed and accuracy.

6. Experiments

Experiment 1: VLAD with FLAIR Speedup In this experiment, we measure the average speed for creating feature vectors for boxes with standard VLAD and VLAD with FLAIR. The implementations run on a single core of a Xeon E3-1270 CPU. The codebook size $K = 256$ and the descriptor has dimensionality $D = 80$, common values in the literature. To measure precisely the speedup provided by FLAIR, the time for finding the closest codewords (0.55s per image) is excluded from our timings.

Figure 3(a) shows the speedup of VLAD with FLAIR for a varying number of boxes. For 30,000 boxes, equivalent to

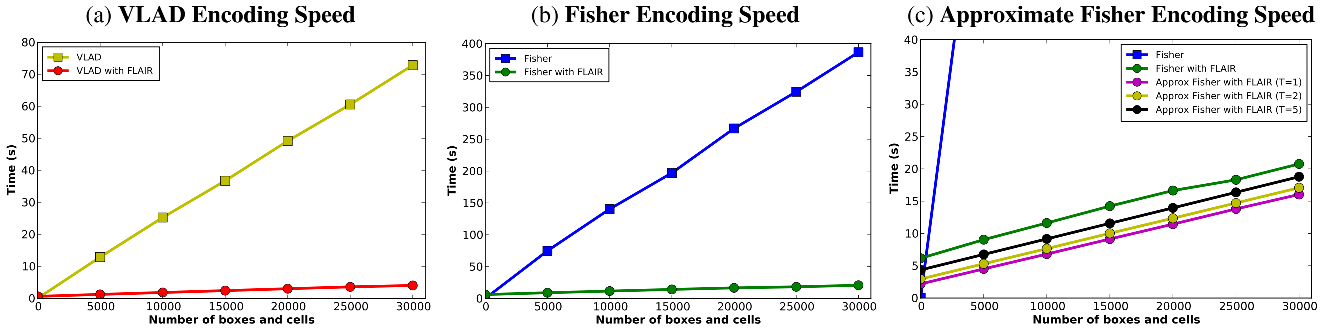

Figure 3. **Experiments 1 and 2: FLAIR speedups** (a) VLAD with FLAIR is 18.0x faster at 30k boxes, (b) Fisher with FLAIR is 18.7x faster at 30k boxes, (c) Approximate Fisher with FLAIR reduces the overhead cost of FLAIR construction for an image.

selective search [24] with a 1x1 and 4x4 spatial pyramid, VLAD with FLAIR is 18.0x faster than standard VLAD: Where regular VLAD requires 73s per image to create feature vectors, VLAD with FLAIR needs only 4.0s. VLAD with FLAIR does have a fixed cost of 0.6s for construction of the representation. However, as soon as the number of boxes to analyze is larger than 150 per image, a number that is easily achieved in practice, it is faster to use VLAD with FLAIR.

Experiment 2: Fisher with FLAIR Speedup Figure 3(b) shows the speedup of Fisher with FLAIR. Again, we exclude the time for codeword assignment (6.2s). For 30,000 boxes, FLAIR is 18.7x faster than standard Fisher: it takes 21s per image to create vectors instead of 6.5 minutes. As soon as the number of boxes is larger than 140, the construction time of Fisher with FLAIR (6.0s) is offset by the more efficient feature vector creation.

Approximate Fisher with FLAIR limits the number of codeword assignments per descriptor to T. This approximation increases the sparsity in the descriptor coordinates and thereby decreases the construction time for FLAIR. Figure 3(c) shows the FLAIR construction timings at the intersection with the Y-axis: for $T = 1$ of 2.2s, for $T = 2$ of 3.0s and for $T = 5$ of 4.4s, compared to 6.0s for the exact version. Approximate Fisher with FLAIR does not reduce the time per box further, but it reduces the fixed overhead cost of FLAIR by several seconds.

Experiment 3: Overall Object Detection Speedup and Accuracy So far our experiments measured the speed of feature encoding with and without FLAIR. However, in a complete object detection pipeline, there are many additional steps: determining boxes to use, extracting descriptors, finding the closest codewords and applying object models. These steps take time and their implementation choice determines the overall accuracy of the object detection pipeline.

Here, we use fast selective search [24] for partitioning the image into bounding boxes. As descriptor we use dense SIFT, sampled at every 2 pixels at 3 scales. The dimensionality of SIFT is reduced to $D = 80$ with PCA. The spatial pyramid is 1x1 and 4x4 after [5]; leaving out the spatial pyramid decreases accuracy by 40%. For each object we train a linear SVM classifier, where the positive examples come from ground truth annotations. The initial set of negative examples are selective search boxes which overlap 20% to 50% with the ground truth boxes. The set of negative examples is extended through hard negative mining [10]: the trained model is applied to the training set, and from each image one box is added to the negative set (provided it does not overlap more than 30% with a ground truth box). We perform two rounds of hard negative mining.

	Time (s)			
	Standard	with FLAIR	Speedup	mAP
BoW [24]	47.9	-	-	32.3
VLAD [14]	34.3	7.8	4.4x	28.2
Fisher [18]	120.0	32.5	3.7x	33.3
Approx Fisher ($T = 1$)	82.2	28.1	2.9x	22.8
Approx Fisher ($T = 2$)	86.0	29.1	3.0x	30.3
Approx Fisher ($T = 5$)	99.8	30.8	3.2x	33.3

Table 1. **Experiment 3: Overall object detection speedup and accuracy for a complete detection pipeline** with 2,000 boxes per image and a spatial pyramid of 1x1 and 4x4, timings per image. With FLAIR, a VLAD pipeline is 4x faster and a Fisher pipeline is 3x faster overall. By approximating Fisher, the time per image is reduced by several seconds, with no loss in mAP for $T = 5$ on the PASCAL VOC 2007 test set. Compared to BoW, Fisher with FLAIR is better and faster.

As a dataset for this comparative experiment of encoding methods we use the PASCAL VOC 2007 dataset with 20 object classes and 10,000 images [9]. To measure accuracy, we use mean Average Precision (mAP) over 20 classes, which is the standard object detection setup on this dataset.

In Table 1, we report the object detection speedup and accuracy. FLAIR makes the entire pipeline 4.4x faster for VLAD, and 3.7x for Fisher: creating the feature vectors for boxes was a substantial computational bottleneck. VLAD is faster than Fisher, partly because the Fisher vector is twice as long. However, in terms of accuracy the Fisher encoding is clearly better: 33.3% mAP versus 28.2% mAP. For reference, a BoW encoding achieves 32.3% mAP. BoW uses a larger codebook size 4,096 and a non-linear Histogram Intersection Kernel, without them the accuracy is significantly lower. Approximate Fisher lowers the execution time per image by several seconds. However, assignment to just 1 or 2 descriptors results in lower accuracy: 22.8% mAP or

System	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
[10]	52.4	54.3	13.0	15.6	**35.1**	54.2	49.1	31.8	15.5	26.2	13.5	21.5	45.4	51.6	**47.5**	09.1	35.1	19.4	46.6	38.0	33.7
[24]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
[5]	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
NLPR	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
[31]	**65.0**	48.9	25.9	24.6	24.5	**56.1**	**54.5**	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	**43.9**	32.6	**54.0**	45.9	39.7
This paper	61.3	52.3	**27.8**	**25.7**	21.3	54.0	45.6	**54.0**	15.5	**32.6**	**33.3**	**41.8**	47.9	**57.8**	37.3	**24.3**	41.8	**35.8**	50.4	**47.3**	**40.4**

Table 2. **Experiment 4: Comparison on the Pascal VOC 2010 detection task**, comparing the approach from this paper to others without context rescoring. We improve the state-of-the-art, and perform best for 9 objects.

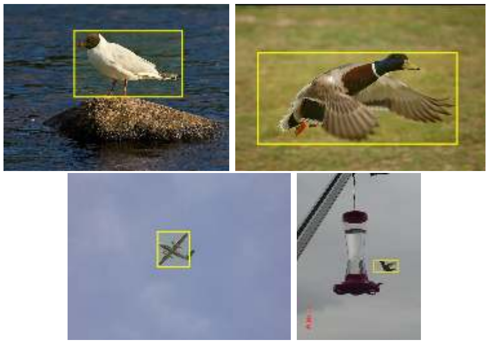

Figure 4. **Qualitative object detection results** on PASCAL VOC 2010 for birds. The airplane on the bottom-left is a false positive.

30.3% mAP. Descriptors should be assigned to 5 codewords to maintain accuracy and still save 1.7s per image compared to exact Fisher with FLAIR. Compared to object detection with BoW, Fisher with FLAIR is better and faster.

Experiment 4: Comparisons to state-of-the-art We now switch to evaluations on the PASCAL VOC 2010 dataset to compare to the state-of-the-art in object detection. Based on the above experiments, we use Fisher with FLAIR due to its accuracy. Color features are commonly used in state-of-the-art systems [5, 24], therefore we include OpponentSIFT and C-SIFT in our system in addition to intensity SIFT. We focus on the box encoding and do not consider post-processing by context rescoring [23, 5]. Because evaluating multiple boxes is computationally cheap in Fisher with FLAIR, we increase the spatial pyramid to 30 cells (1x1, 2x2, 3x3 and 4x4) from the previously used 17. In Table 2, we compare our results to 5 representative state-of-the-art detectors. By exploiting the speedup offered by Fisher with FLAIR for inclusion of color and fine spatial pyramids we obtain the best results for 9 objects and the best overall mAP. This system does not use any context rescoring. We highlight qualitative detection results for the bird category in Figure 4. We also submitted the same system to the online leaderboards of the VOC 2012 dataset (40.6 mAP); detailed results are available online.

We also evaluate Fisher with FLAIR on the fine-grained species categorization task specified in the CUB-2011 dataset containing two hundred birds [30]. For each of the bird species there are 30 training images and 30 testing images. We use the standard training/test split provided by the authors. Following the standard evaluation protocol, we mirror the train images to double the size of the training set. Note that in this task one should evaluate for uncropped images, similar to [32], without using the provided box annotations at test time. We do not use any bird-specific optimizations, we just rely on the same implementation as before, but without C-SIFT. We report the standard evaluation metric, that is the mean accuracy over all the 200 species, in Table 3. Compared to the baseline and the best known number in the literature, we nearly double the mean accuracy. A considerable improvement. Fisher with FLAIR opens doors to fine-grained object detection. Although popular, we consider using the human provided bounding boxes of the CUB-2011 test set to be an unrealistic scenario. As in practice these bounding boxes will not be availabe. With these bounding boxes our accuracy increases to 55.5% where [2] achieve 56.8% with boxes. Please note, in contrast to [2] who use domain-specific knowledge by design, we use no form of domain-specific feature optimization. Our fine-grained categorization results from generic object detection, which also detects fine-grained objects when bounding box annotations on the test set are absent (!). Fisher with FLAIR opens doors to fine-grained object detection in real-world scenarios.

System	Accuracy	
	No test boxes	With test boxes
[30]	10.3	17.3
[32]	28.2	-
[2]	-	56.8
This paper	52.2	55.5

Table 3. **Experiment 4: Comparison on the CUB-200-2011** bird species categorization task, comparing our Fisher with FLAIR to the state-of-the-art. We improve the state-of-the-art by over 20% in mean accuracy for the real-world task where there are no ground truth bounding boxes on the test set. Fisher with FLAIR opens doors to fine-grained object detection without domain-specific feature optimization.

Finally, we evaluate Fisher with FLAIR on the ImageNet 2013 detection challenge over 200 classes. Results for the validation set and a comparison with other methods is shown in Table 4. In the ImageNet 2013 challenge, Fisher with FLAIR was the number one system submitted, achieving 22.6 mAP on the test set.

7. Conclusion

FLAIR is the new data representation which enables fast encoding of arbitrary boxes in an image with the powerful VLAD and Fisher vectors. FLAIR splits an image over binary and integral images, one per code value in the codebook. Once an image is represented in FLAIR, the evaluation of VLAD and Fisher is independent of the area of the

System	mAP
[10] from [31]	10.0
[31]	14.7
This paper	18.3
This paper (with context)	21.9

Table 4. **Experiment 4: Comparison on the ImageNet 2013** detection challenge validation set over 200 categories. On the test set, Fisher with FLAIR was the best system submitted to the challenge with 22.6 mAP.

box, and restricted to four additions. The method rests on the linear sum over the pixels of these methods, but an extension of FLAIR allows also for the sum of squares for the important ℓ_2-norm.

FLAIR can be used for the gain in speed as we demonstrate in this paper to be a factor 18 in the state of-the-art VLAD [14] and Fisher vector [18, 22] encodings. By evaluating a large number of relevant boxes, selected by selective search [24], we set a new state-of-the-art for object detection on the challenging PASCAL VOC 2010 and achieve the top rank in the ImageNet 2013 detection challenge. Alternatively, FLAIR can be used to increase the number of boxes in an evaluation, which would otherwise be prohibited by the computational effort. For fine-grained object categorization and localization we are able to evaluate 2,000 boxes and still use 30 pyramids in each box as the evaluation of many boxes is cheap in FLAIR. As a result, without any further knowledge of the birds [30], their pose [32] or location [2] we obtain an accuracy of 55.5%.

We conclude that the computational efficiency of FLAIR opens the door to modern yet computationally expensive techniques for object categorization and localization in large image collections, video archives, as well as for mobile visual recognition.

Acknowledgments This research is supported by the STW STORY project and the Dutch national program COMMIT.

References

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

[2] T. Berg and P. N. Belhumeur. POOF: Part-Based One-vs-One Features for fine-grained categorization, face verification, and attribute estimation. In *CVPR*, 2013.

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

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

[5] R. G. Cinbis, J. Verbeek, and C. Schmid. Segmentation driven object detection with fisher vectors. In *ICCV*, 2013.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2009.

[7] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Statistical Learning in Computer Vision*, 2004.

[8] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *CVPR*, 2013.

[9] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010.

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 32:1627–1645, 2010.

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

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

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

[14] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *TPAMI*, 34(9):1704–1716, 2012.

[15] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *TPAMI*, 31(12):2129–2142, 2009.

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

[17] V. Lempitsky and A. Zisserman. Learning to count objects in images. In *NIPS*, 2010.

[18] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[19] F. M. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *CVPR*, 2005.

[20] E. Rahtu, J. Kannala, and M. Blaschko. Learning a category independent object detection cascade. In *ICCV*, 2011.

[21] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei. Object-centric spatial pooling for image classification. In *ECCV*, 2012.

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

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

[24] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[25] K. E. A. van de Sande, J. R. R. Uijlings, C. G. M. Snoek, and A. W. M. Smeulders. Hybrid coding for selective search. In *ECCV PASCAL VOC*, 2012.

[26] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. Visual word ambiguity. *TPAMI*, 32(7):1271–1283, 2010.

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

[28] S. Vijayanarasimhan and K. Grauman. Efficient region search for object detection. In *CVPR*, 2011.

[29] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[30] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical report, 2011.

[31] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *ICCV*, 2013.

[32] N. Zhang, R. Farrell, and T. Darrell. Pose pooling kernels for subcategory recognition. In *CVPR*, 2012.

[33] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classiffication using super-vector coding of local image descriptors. In *ECCV*, 2010.