UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Codemaps

Segment, Classify and Search Objects Locally

Li, Z.; Gavves, E.; van de Sande, K.E.A.; Snoek, C.G.M.; Smeulders, A.W.M.

DOI
10.1109/ICCV.2013.454

Publication date
2013

Document Version
Author accepted manuscript

Published in
2013 IEEE International Conference on Computer Vision

Link to publication

Citation for published version (APA):
Li, Z., Gavves, E., van de Sande, K. E. A., Snoek, C. G. M., & Smeulders, A. W. M. (2013). Codemaps: Segment, Classify and Search Objects Locally. In *2013 IEEE International Conference on Computer Vision: ICCV 2013 : proceedings: 1-8 December 2013, Sydney, NSW, Australia* (pp. 2136-2143). IEEE Computer Society. https://doi.org/10.1109/ICCV.2013.454

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:21 Mar 2026

Codemaps
Segment, Classify and Search Objects Locally

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

Abstract

In this paper we aim for segmentation and classification of objects. We propose codemaps that are a joint formulation of the classification score and the local neighborhood it belongs to in the image. We obtain the codemap by reordering the encoding, pooling and classification steps over lattice elements. Other than existing linear decompositions who emphasize only the efficiency benefits for localized search, we make three novel contributions. As a preliminary, we provide a theoretical generalization of the sufficient mathematical conditions under which image encodings and classification becomes locally decomposable. As first novelty we introduce ℓ_2 normalization for arbitrarily shaped image regions, which is fast enough for semantic segmentation using our Fisher codemaps. Second, using the same lattice across images, we propose kernel pooling which embeds nonlinearities into codemaps for object classification by explicit or approximate feature mappings. Results demonstrate that ℓ_2 normalized Fisher codemaps improve the state-of-the-art in semantic segmentation for PASCAL VOC. For object classification the addition of nonlinearities brings us on par with the state-of-the-art, but is 3x faster. Because of the codemaps' inherent efficiency, we can reach significant speed-ups for localized search as well. We exploit the efficiency gain for our third novelty: object segment retrieval using a single query image only.

1. Introduction

It remains remarkable that the great successes in object recognition use so little of the spatial order in the image. Features [20] are encoded in feature space [14, 23, 26, 35], pooled in histograms [4, 35] and plugged into a kernel-classifier [7, 36]. The entire chain contains no more spatial information then the locality of the features, compensated by a rather crude method of spatial pyramids [12, 18] where the standard classification procedure is repeated over upper and lower parts of the image. To make progress in recognition, better inclusion of more spatial information is inevitable. And indeed, recently, [13, 28, 29] have introduced the spatial coherence of objects, by confining the usual analysis to selected windows in the image. The spatial restriction has had a positive effect on the recognition result. However, the selection of windows is only loosely coupled to the classification pipeline. In this paper, we aim to integrate locality much further into the analysis.

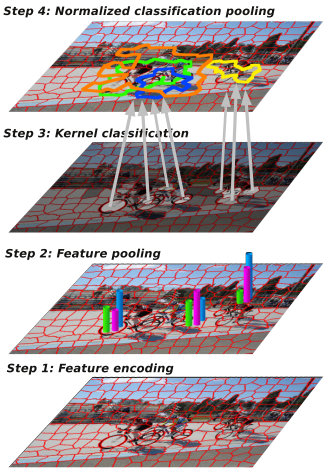


Figure 1. **Codemaps** segment, classify and search objects locally by reordering the encoding, pooling and classification steps of object classification. Different from existing linear decompositions for specific pipelines, codemaps are generic, embed fast ℓ_2 normalization, include nonlinearities by local kernel pooling and allow for segment retrieval using a single query image only.

Starting at the other end, the route of first segmentation then recognition, is as old as Blobworld [6], describing parts as visually coherent regions. In [16, 33] the regions were jointly modeled to establish semantic similarity between adjacent object parts. And indeed, consistent regions lead to useful object hypotheses [1, 5], paving the way to object

segmentation. Pushing localization with state-of-the-art encodings to an extreme, [8] classifies pixels with a Fisher kernel for semantic segmentation, as application of Fisher on regions would be practically infeasible. We will argue in this paper the virtue of a deeper connection between spatial localization and object type classification using state-of-the-art pipelines, such as the improved Fisher kernel [23] or explicit feature maps [32].

We aim to combine semantic segmentation with recognition at the earliest stage of the analysis. We show that reordering the processing steps for object type classification into local pooling before classification has considerable advantages. Where [17, 34] have shown the efficiency benefits of such decompositions for unnormalized bag-of-words with a linear classifier, our *codemaps* make three novel contributions. As a preliminary result, we formulate the sufficient mathematical conditions under which image encoding and classification are locally decomposable (Section 3). In the first novelty, we use this result to introduce codemaps with ℓ_2 normalization for arbitrarily shaped image regions (Section 4), essential to reach a better than state-of-the-art performance in semantic segmentation [1, 5]. In the second novelty, using the same lattice across images, we include nonlinearity in the decomposition by local kernel pooling (Section 5), to bring us on par with the state-of-the-art in object classification [23], but 3x faster. Thirdly, we demonstrate the effectiveness of codemaps in object segment retrieval from a single query image (Section 6).

2. Related work

We structure our discussion on related work by the subsequent steps of (localized) object type segmentation and classification: semantic segmentation, feature encoding, feature pooling and kernel classification.

Semantic segmentation. For semantic segmentation two main approaches have been adopted, which both start from an initial image split into superpixels. The first approach [16, 33] then tries to group the superpixels on the basis of semantic similarity in a conditional random field. Such approaches achieve excellent accuracy, but suffer from a high computational cost due to complicated training and inference. The second approach [1, 5] first tries to form complete segment hypotheses based on low level cues. Then, these segment hypotheses are classified with a standard object classification procedure. In the current work we follow the second family of approaches. We take advantage of the complete segment hypotheses being composed of superpixels to enrich the segment representation with state-of-the-art image classification using feature encodings.

Feature encoding. Feature encodings capture the visual information around a local neighborhood and generate a measurement, which is supposed to be invariant to accidental circumstances, such as illumination, shade, occlusion etc. A feature encoding uses the raw pixel data from a local neighborhood to generate a code [20, 30]. The acquired code is projected to a, usually, higher dimensional space. The most popular projections are vector quantization [26], soft quantization [31, 35], or the difference of projections to pre-trained models captured by Fisher [23] and VLAD [14] vectors. The framework we propose is not constrained to any particular encoding choice, as long as they are local. In our experiments we use Fisher vectors, as they have shown to yield state-of-the art results in object classification [23] and retrieval [14].

Feature pooling. The feature pooling spatially aggregates the relevant local feature encodings into a global image representation. Average pooling has been shown to work best for bag-of-words [26] and Fisher vectors [23]. Max-pooling is proven effective for sparse coding [35] and deep learning [15]. Sum pooling has been the preferred choice for VLAD vectors [14]. In this paper we generalize on the pooling functions. We show that pooling over a region of interest is equivalent to a simpler two-level pooling for a particular family of mathematical functions. This two-level pooling allows to classify objects locally, while offering a substantial efficiency speed-up.

Kernel classification. For object type classification, support vector machines have repeatedly shown to outperform [9, 11, 27] all other alternatives. To cope with the growing number of images, the size of the image representations and the numbers of object types, recent research has focused on efficient learning and classification. Kernel properties, like additivity and homogeneity, of support vector machines have been exploited for speed-ups, especially for nonlinear kernels [21, 32]. Classifiers currently employ the local origin of the data only weakly [18, 29]. In this work we modularize linear and nonlinear kernels to arrive at an object type decision for a local neighborhood level.

We are inspired by [17, 34] who observe that the image interpretation of unnormalized bag-of-words with a linear classifier can be analyzed in terms of the contributions of individual descriptors, leading to a considerable efficiency gain. They don't seem to realize that the decomposition can be generalized, as we do, to reorder the steps of general object classification pipelines, including for example Fisher vectors and VLAD. By doing so, we obtain a joint formulation of the classification score and the local neighborhood it belongs to. Furthermore, our generalized framework can obtain the precise ℓ_2 normalized classification score for any region, which is known to increase the accuracy of both Fisher vectors and VLAD considerably [14, 23]. Last but not least, we propose kernel pooling which embeds nonlinearities by explicit or approximate feature mappings [21, 32] to assure state-of-the-art competitiveness [23]. We call our approach *codemaps* and we will now highlight its theoretical foundation in a preliminary.

3. Preliminary

We start from a lattice, composed of N nodes, $G = \{g_i\}, i = 1, ..., N$, superimposed on an image. To ensure good generalization and flexibility, we consider that *a)* each node g_i of the lattice is arbitrarily sized, shaped, and non-overlapping, *i.e.* $g_i \cap g_j = \varnothing, \forall g_i, g_j \in G, i \neq j$, and *b)* each area R where we search for the objects of interest are composed of multiple nodes $R = g_1 \cup ... \cup g_l$. Thus, regions are also arbitrarily sized and shaped. Hence, the image search is no longer confined to specific and limiting templates, such as rectangular areas [17, 29]. For ease of reading we shall refer to each node g_i of the lattice as *lex*. Our theory holds for all types of patches, including cells on a regular lattice [18], generalized image regions [2], superpixels [19, 24] or any other type of localities.

We extract a collection of local features $z_i, i = 1, ..., M$ in the image and encode them to codes $c_i, 1, ..., M$. The pooling function $h(R)$ combines these local codes within the region R to arrive at its global feature encoding.

Codemaps. We define a *codemap* as a decomposed object image representation $\Phi = (G, \phi)$, where $\phi_i \in \phi, i = 1, ..., N$ denotes the object evidence per lex g_i.

We begin with unnormalized codemaps. Hence, $\phi_i = f(h(g_i))$ stands for the per lex classification score using classifier function f. For an image region R the corresponding classification score can be written as:

$$f(h(R)) = f(h(g_1 \cup g_2 \cup ... \cup g_l)). \tag{1}$$

Formally, the property of a codemap can be described by

$$f(h(R)) = q(f(h(g_1)), f(h(g_2)), ..., f(h(g_l))), \tag{2}$$

where q is a classification pooling function that aggregates the localized classifier decisions over a region of interest.

From eq. (2) we see that the pooling function h needs to be applied to each of the lexes g_i separately. Taking into account eq. (1), we arrive at the first condition for obtaining a valid codemap:

Condition 1 *The pooling function $h : A \rightarrow B$ must be homomorphic from the space A to space B, so that*

$$\begin{aligned} h(R) &= h(\mathcal{U}_A\Big[g_1, g_2, ..., g_l\Big]) \\ &= \mathcal{U}_B\Big[h(g_1), h(g_2), ..., h(g_l)\Big] \end{aligned} \tag{3}$$

where A refers to the spatial domain formed by lexes $\{g_i\}$, and B refers to the code pooling space defined by h. In eq. (3) the $\mathcal{U}_A$ stands for the union set operation, that is $\mathcal{U}_A = \cup(g_1, g_2, ..., g_l)$, whereas $\mathcal{U}_B$ is an operation in B that makes h homomorphic. When h stands for sum pooling or max pooling, $\mathcal{U}_B$ is the sum operator or max operator. In practice a homomorphic pooling means that we can first locally pool the encodings from each lex g_i separately, then combine them to get the global feature encoding as if we operated on R in the first place.

In addition, we want the classifier f to also operate on each of the lexes g_i individually. By combining eq. (1) and (3), we arrive at the second condition:

Condition 2 *The classification function $f : B \rightarrow C$ must be homomorphic from the space B to space C, so that*

$$\begin{aligned} f(h(R)) &= f(\mathcal{U}_B\Big[h(g_1), h(g_2), ..., h(g_l)\Big]) \\ &= \mathcal{U}_C\Big[f(h(g_1)), f(h(g_2)), ..., f(h(g_l))\Big] \end{aligned} \tag{4}$$

where C refers to the classification space. Having a homomorphic function for the classifier f, one only needs to consider the individual scores of the lexes within R.

Normally, when classifying a region we first perform a global pooling on all the feature encodings contained in the region, and then we apply the classifier. However, according to Cond. 1, codemaps first break the global pooling into a collection of local feature poolings over lexes. Then, according to Cond. 2, codemaps apply the classifier on the local feature poolings and perform a global pooling on the classification scores of the lexes. Hence, the global pooling is performed on single scalars instead of high dimensional vectors. This brings significant efficiency benefits for vision tasks where thousands of regions need to be classified per image, such as in semantic segmentation.

We conclude that if Cond. 1 and 2 are met, we obtain a valid codemap.

4. ℓ_2 normalization for arbitrary regions

Modern feature encodings, such as Fisher vector, VLAD or bag-of-words, usually include a summation operator in the feature pooling function h. When a linear classifier f is used, the classification score of a region is $y(R) = \mathbf{w}^T h(R) = \sum_d \sum_{g_i \in R} w_d h_d(g_i)$, where $\mathbf{w}$ denotes the learned d dimensional weights by the linear classifier. This leads to a valid codemap, since

$$y(R) = \sum_{i=1}^{l} y(g_i), \tag{5}$$

where $y(g_i) = \sum_d w_d h_d(g_i)$. A similar decomposition was derived in [17, 34] for the specific case of unnormalized bag-of-words with a linear SVM. However, feature encodings also profit highly from normalization before classification [23, 32]. Including normalization, in particular ℓ_2, for variable spatial regions is difficult to do efficiently. We propose ℓ_2 normalization for arbitrary regions in codemaps.

In general for a region R the norm of its feature encoding vector $h(R)$ is $\|L_R\|$. Because of the linear classifier we can rewrite the normalized classification score as:

$$y(R) = f(\frac{1}{\|L_R\|} \cdot h(R)) = \frac{1}{\|L_R\|} \cdot f(h(R)). \quad (6)$$

As eq. (6) indicates, to obtain the normalized classification score we can postpone the scaling by the inverse norm $\frac{1}{\|L_R\|}$ until after the classification pooling. Thus, with codemaps, normalization boils down to multiply by a scalar on the classification score of a region instead of the high dimensional feature encodings. Then we consider the ℓ_2 norm of the feature encoding for a region R within an image, since the linear classifier prefers ℓ_2 normalization. It is equal to the square root of the dot product of $h(R)$ with itself:

$$\begin{aligned} \|L_R\|_2 &= \left(h(R)^T h(R)\right)^{1/2} \\ &= \left(\sum_{i=1}^{l}\sum_{j=1}^{l} h(g_i)^T h(g_j)\right)^{1/2}. \end{aligned} \quad (7)$$

As we can see from eq. (7), to calculate the ℓ_2 norm of a region R we only need to know the sum of the pair-wise dot product $h(g_i)^T h(g_j)$ between feature encodings of the lexes within the region. To generalize for any arbitrary region R, we calculate the dot products of all the pair-wise lex combinations in the image. Then, we only need to consider the combinations of lexes that both appear in R, that is:

$$\|L_R\|_2 = \left(\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\, \alpha_j h(g_i)^T h(g_j)\right)^{1/2}, \quad (8)$$

where the binary vector $\alpha = (\alpha_1, ..., \alpha_N)$ indicates whether each lex is present or not within the region R. Finally, we compute the ℓ_2-normalized classification score of an arbitrary region R as:

$$y(R) = \frac{1}{\|L_R\|_2} \sum_{i=1}^{N} \alpha_i \mathbf{w}^T h(g_i). \quad (9)$$

We describe the ℓ_2 normalized codemap of an image as:

$$\Phi = \{g_i, \mathbf{w}^T h(g_i), h(g_i)^T h(g_j)\}, \quad (10)$$

for $i, j = 1, ..., N$.

Fisher codemaps. The popular Fisher vector, extracted from a Gaussian mixture model with a probability density function $u(; |\boldsymbol{\mu}, \boldsymbol{\sigma})$ is equal to $c_{z_i} = \frac{1}{M_R} \bigtriangledown_{\boldsymbol{\mu},\boldsymbol{\sigma}} \log u_{\boldsymbol{\mu},\boldsymbol{\sigma}}(z_i)$, where M_R stands for the number of local descriptors z_i sampled from an image. A codemap is independent of the regions R, hence the value of M_R is not available. However, M_R is canceled out during the ℓ_2 normalization, therefore we propose to drop the constant M_R from the original Fisher vector by $\widetilde{c}_{z_i} = \bigtriangledown_{\boldsymbol{\mu},\boldsymbol{\sigma}} \log u_{\boldsymbol{\mu},\boldsymbol{\sigma}}(z_i)$. Since we use the sum operator for the feature pooling and the sum operator due to the linear classifier, the Cond. 1 and 2 are fulfilled and we obtain a valid *Fisher codemap*. We observe in Figure 2(a) that Fisher codemaps allow for a considerable speed-up when classifying a number of arbitrary sized, shaped regions. For evaluating 1,000 regions the Fisher codemap needs 23 seconds per image, as compared to 22 minutes when using the traditional Fisher vectors. The speed-up improves further when more evaluations are required. Moreover, for a large number of classes, codemaps still have a clear advantage. The computation of the normalization matrix $h(g_i)^T h(g_j), i, j = 1, ..., N$ for codemaps is shared across all the object classes. Therefore, for classifying 1,000 object categories over 1,000 image regions, the Fisher codemaps are still 45x faster than the Fisher vectors. We conclude that our ℓ_2 normalized Fisher codemaps are mathematically equivalent to Fisher vectors, but much faster. Similar formulations and efficiency benefits can be derived for other feature encodings, *e.g*. bag-of-words or VLAD, as well.

5. Nonlinear kernel pooling for classification

In principle, codemaps work with arbitrary lattices for images. We now approach codemaps from a kernel point of view, aiming to introduce nonlinearities for object classification using the same lattice across images. Given two codemaps Φ_X and Φ_Z for image X and Z, the similarity between two images using a linear kernel is:

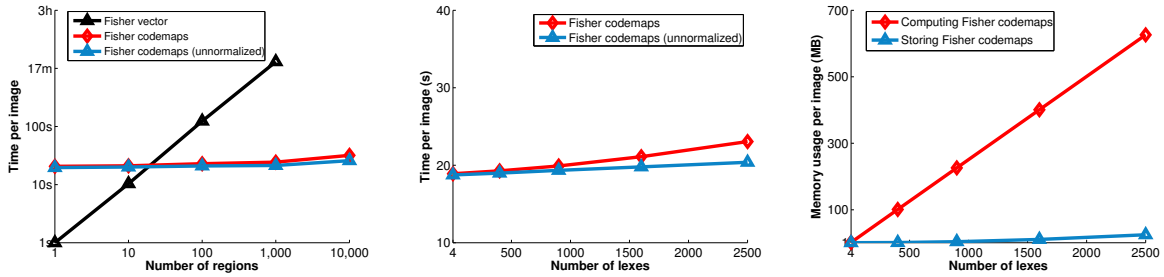
$$\begin{aligned} K_L(X, Z) &= h(X)^T h(Z) \\ &= \sum_{g_x \in X} \sum_{g_z \in Z} h(g_x)^T h(g_z), \end{aligned} \quad (11)$$

which is equivalent to the sum of the similarities using linear kernel between pair-wise lexes. Hence, we can apply the kernel trick and use more sophisticated kernels to measure the pair-wise lex similarity. In that case the similarity between images X and Z becomes:

$$\widetilde{K}(X, Z) = \sum_{g_x \in X} \sum_{g_z \in Z} k(h(g_x), h(g_z)). \quad (12)$$

$\widetilde{K}(X, Z)$ is a positive definite kernel as long as k is positive definite as well [25]. This is related to match kernels [3] between sets of local features, *e.g*. SIFT, but we consider kernels between lexes. All the standard kernels from the literature are applicable for k in eq. (12). In order to preserve Cond. 1 and 2 of codemaps, we opt for explicit or approximate kernel mappings [21, 25, 32], that is:

$$\begin{aligned} \widetilde{K}(X, Z) &= \sum_{g_x \in X} \sum_{g_z \in Z} \psi(h(g_x))^T \psi(h(g_z)) \\ &= \widetilde{h}(X)^T \widetilde{h}(Z), \end{aligned} \quad (13)$$



(a) (b) (c)

Figure 2. **Timing and memory usages for codemaps.** All the experiments are done on the PASCAL VOC images using a Gaussian mixture model of 256 components. (a) ℓ_2 normalized Fisher codemaps (with 400 lexes per image) are up to 56x faster than traditional Fisher vectors, depending on the number of regions analyzed (note the $\log 10/\log 10$ scales). (b) ℓ_2 normalization for arbitrary regions is efficient. For the $4-500$ lexes per image that usually suffice for semantic segmentation [2, 5, 33], the unnormalized and the normalized Fisher codemaps are practically as efficient, but the normalized Fisher codemaps are much more effective as shown in Table 1. (c) Depending on the number of lexes, computing Fisher codemaps costs up to 600 MB memory per image, while storing them only needs less than 30 MB.

where $\widetilde{h}(X) = \sum_{g_x \in X} \psi(h(g_x))$ indicates the nonlinear feature pooling function for image X, which is the sum of the set of pooled lexes applied by nonlinear kernel mapping ψ. We refer to $\psi(h(g_i))$ as local nonlinear kernel pooling. Thus we still use the sum operator for global feature pooling and a linear classifier, which lead to a valid codemap. For normalization we need $\widetilde{K}(X, X) = 1$, which is equivalent to use ℓ_2 normalization on $\widetilde{h}(X)$. Then the resulting codemap with nonlinear kernel pooling is defined as:

$$\Psi = \{g_i, \mathbf{w}^T \psi(h(g_i)), \psi(h(g_i))^T \psi(h(g_j))\}, \qquad (14)$$

for $i, j = 1, ..., N$.

Applying nonlinear kernel pooling for each lex makes the global image feature encoding $\widetilde{h}(X)$ dependent on the partition of the lattice elements placed on the image. Therefore, to ensure a consistent image representation so that we can measure image similarity properly, we define the same lattice across all the images. Consequently, codemaps with kernel pooling have a strong connection with spatial pyramid kernels. For the spatial pyramid kernel we compute the similarity of each lex in an image only with itself, whereas for codemaps all the pair-wise similarities between lexes are considered. Hence, one could view our codemaps with kernel pooling as an extension of the spatial pyramid kernels. However, for our kernel pooling the final classification score is computed from a single lattice based on all the partitions of spatial pyramids without any redundancy, where spatial pyramids require multiple layouts. As a result, codemaps with kernel pooling allow for the inclusion of richer spatial information in the final classification score at a nearly zero cost.

6. Experiments

We demonstrate the efficiency and effectiveness of proposed codemaps by experiments on semantic segmentation, object classification and segmented object retrieval. Since these tasks all require repetitive computations on overlapping regions, performing them once with ℓ_2 normalized codemaps and nonlinear kernel pooling leads to a considerable speedup. We summarize the timing and memory usages in Figure 2.

	Bag-of-words	Fisher codemaps	
Region normalization	–	–	ℓ_2
mAP	4.1	7.0	26.9

Table 1. **ℓ_2 normalized semantic segmentation** for arbitrary regions is effective. Mean average precision on the *val* set of the segmentation task in the PASCAL VOC 2011, where – indicates unnormalized versions over regions.

6.1. ℓ_2 normalized semantic segmentation

In the first experiment we quantify the value of codemaps with ℓ_2 normalization for semantic segmentation, where several image regions need to be evaluated on presence of objects and their type. We use the PASCAL VOC Segmentation dataset and follow the training protocol of *CPMC-O_2P* [5], which combines three segmentation-tailored and color-enhanced features, and trains linear support vector regressors based on the overlaps between segments. We use the Fisher codemaps from Section 4, with dense sampling of basic intensity SIFT descriptors per pixel at multiple scales and a Gaussian mixture model of 128 components. Note that we do not consider any feature-specific optimizations for the purpose of semantic segmentation. We also consider the unnormalized Fisher codemap version and unnormalized bag-of-words features using a visual codebook of size 4,000, similar to the ones used in [34]. While unnormalized Fisher vectors have been used for pixel-level segmentation [8], we are unaware of segment-level classification

	mAP	Bgnd	Plane	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	House	M/bike	Person	P/Plant	Sheep	Sofa	Train	TV
CPMC-O_2P [5]	46.4	84.7	63.5	23.4	45.0	40.8	44.9	59.1	58.3	**57.1**	11.8	42.9	32.8	**45.2**	55.4	56.6	51.2	35.6	44.9	30.3	48.0	42.5
FGT_SEGM [10]	47.5	85.2	63.4	**27.3**	**56.1**	37.7	47.2	57.9	59.3	55.0	11.5	**50.8**	30.5	45.0	58.4	57.4	48.6	34.6	**53.3**	32.4	47.6	39.2
DivMBest+Rerank [22]	48.1	**85.7**	62.7	25.6	46.9	**43.0**	**54.8**	58.4	58.6	55.6	**14.6**	47.5	31.2	44.7	51.0	60.9	53.5	**36.6**	50.9	30.1	**50.2**	**46.8**
[5]+*Fisher codemaps*	**48.3**	85.3	**66.2**	24.4	47.5	37.2	52.4	**60.4**	**61.1**	56.5	12.8	44.5	**32.9**	44.8	**60.8**	**61.3**	**55.8**	33.2	49.8	**34.3**	47.9	45.0

Table 2. **State-of-the-art semantic segmentation.** Following the exact protocol of [5] we show semantic segmentation results for the PASCAL VOC 2012 *comp6* task. Adding normalized Fisher codemaps on top of the *CPMC-O_2P* improves the state-of-the-art in semantic segmentation for 8 out of 21 object categories. Best result denoted in bold.

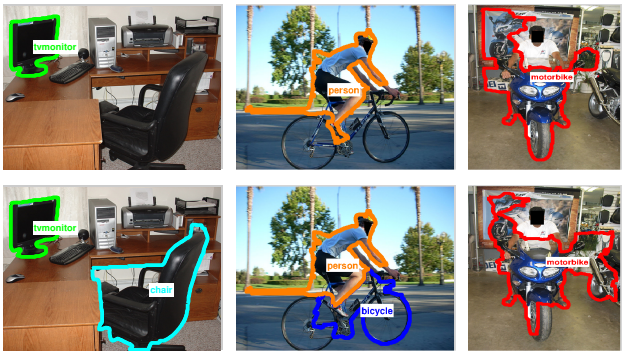


Figure 3. **Semantic segmentation.** Adding normalized Fisher codemaps (bottom row) on top of the *CPMC-O_2P* [5] (top row) appears to be beneficial when multiple objects appear simultaneously in the image. Note for example the difficult case in the last column, where codemaps help better segmenting the motorbike on the poster and the motorbike in the right part of the image.

with normalized Fisher vectors.

Fisher codemaps. We first consider the benefit of ℓ_2 normalization. We use the VOC 2011 *train* set for training and we report results on the *val* set in Table 1. We observe that ℓ_2 normalized Fisher codemaps outperform the unnormalized ones by far. Fisher codemaps obtain a 26.9 mAP (mean Average Precision), where the unnormalized Fisher codemaps obtain only 7.0 mAP. While unnormalized Fisher codemaps outperform bag-of-words, the ℓ_2 normalization is critical for linear regression, since we have to ensure that the overlap between each segment and itself is largest and equal to 1. We plot in Figure 2(b) how efficient it is to compute a Fisher codemap, under a varying number of lexes in the lattice. Calculating the normalized Fisher codemap is as efficient as the unnormalized version for up to 500 lexes. For semantic segmentation in particular, since $4-500$ lexes per image usually suffice [2, 5, 33], calculating the ℓ_2 normalized Fisher codemaps is practically as efficient as the unnormalized one, but much more accurate.

CPMC-O_2P + Fisher codemaps. Since the leading semantic segmentation methods use multiple features to capture several aspects of the object information, *i.e.* in [5] 3 features and in [1] 58 features are used, we embed Fisher codemaps into the multi-feature approach of *CPMC-O_2P* [5] to improve the state-of-the-art in semantic segmentation. We note that the individual features of *CPMC-O_2P*, *i.e.* eSIFT, eMSIFT and eLBP, obtain 28.4, 31.0 and 21.2 mAP on the VOC 2011 *val* set respectively, where Fisher codemaps score 26.9 without any optimizations for semantic segmentation. In this experiment we use the additional training set, the same as *CPMC-O_2P* [5], and report the results on *comp6* of the VOC 2012 challenge. Since both the features in [5] and ℓ_2 normalized Fisher codemaps use a linear regressor, we rely on late fusion with linear weights learned on the *val* set to combine them. We also compare against the best reported methods so far [10, 22]. The numerical results are shown in Table 2. Adding Fisher codemaps brings more precision to the image region representations. More specifically, the semantic segmentation accuracy is improved for 8 out of the 21 object categories (including "background"). In Figure 3 we illustrate some of the segmentation results. We observe that Fisher codemaps are particularly helpful when multiple objects are present simultaneously. We conclude that a combination of *CPMC-O_2P* with our ℓ_2 normalized Fisher codemaps improves the state-of-the-art in semantic segmentation.

6.2. Nonlinear kernel pooling for classification

In the second experiment we quantify the value of codemaps with ℓ_2 normalization and nonlinear kernel pooling for object classification. We use the PASCAL VOC 2007 Classification dataset [11] for both bag-of-words and Fisher vectors [23]. We sample dense SIFT descriptors every two pixels at multiple scales. We use a visual codebook of size 4,000 with hard assignment for the bag-of-words and a 256 component Gaussian mixture model for the Fisher vectors. We employ 1x1, 2x2 and 3x1 spatial pyramids. Since power normalization has shown to work particularly well for Fisher vectors [23], we implement Fisher codemaps with local Hellinger kernel pooling. For bag-of-words the χ^2 and histogram intersection kernels are the top performers [21, 36]. We therefore implement bag-of-words codemaps with local χ^2 and histogram intersection kernel poolings using explicit feature maps [32]. While both behave similarly compared to normal bag-of-words, we report only the slightly better performing histogram intersection kernel. The numerical results are shown in Table 3.

For both bag-of-words and Fisher, a codemap with ℓ_2 normalization is mathematically equivalent to the regular ℓ_2 normalized linear models. Hence the results for the linear

	Bag-of-words		Fisher	
	Others	*Codemaps*	*Others*	*Codemaps*
ℓ_2 **normalization**	42.4	42.4	55.0	55.0
Nonlinearities	54.9	54.8	61.6	61.6

Table 3. **Nonlinear kernel pooling for object classification.** Mean average precision on the PASCAL VOC 2007, following the protocol in [23]. Both bag-of-words and Fisher codemaps with the proposed ℓ_2 normalization and nonlinear kernel pooling have the same accuracy as the state-of-the-art. Where the best Fisher vectors require 18 seconds per image for evaluating all 20 classes, our codemaps require only 6 seconds.

classifier are identical. When we add nonlinearities to the bag-of-words by means of a histogram intersection kernel the results increase from 42.4 to 54.9 mAP. With histogram intersection kernel pooling using approximate feature maps we obtain practically the same result for codemaps: 54.8 mAP. Fisher vectors outperform the bag-of-words to a maximum results of 61.6 mAP using ℓ_2 normalization and a power norm nonlinearity. With Hellinger kernel pooling we reach the same result. However, our codemaps only need a single-resolution lattice, as compared to the multiple lattices required by spatial pyramid kernels. Hence, we can evaluate an image for all 20 classes in 6 seconds, where Fisher vectors require 18 seconds. Since it also costs around 6 seconds for Fisher vectors to test an image without any spatial pyramids, codemaps can include full spatial pyramids with nearly zero additional cost, but increase the mAP from 57.4 to 61.6. Moreover, in 18 seconds per image we can also embed RGB-SIFT and OpponentSIFT from [30] in a colored Fisher codemap, by simple average fusion resulting in 64.1 mAP. Codemaps with our proposed ℓ_2 normalization and nonlinear kernel pooling are as good as the state-of-the-art, but 3x more efficient to compute.

6.3. Codemaps for segmented object retrieval

In the last experiment we take advantage of the efficiency benefits of ℓ_2 normalized codemaps to revisit the old challenge of object segment retrieval [6] and we suggest a new solution. We propose to apply codemaps for segmented object retrieval in a query-by-example setting. We use the query images from the instance search task of the TRECVID 2012 [27] video retrieval benchmark. Note that we do not intend to embed in the regular setting of full image instance search, but to explore whether image regions can be retrieved using a single query image only. We extract normalized Fisher codemaps in the same way as the previous experiment. As lexes we use the superpixel regions from [2].

We use the feature encoding of the segmented query as a classifier function. For retrieval, the cosine similarity is measured by a dot product between the ℓ_2 normalized query and the retrieved images. We proceed on searching for those groups of lexes that maximize this cosine similarity. For the search, we devise a simple greedy algorithm. We first look for those lexes most similar to the segmented query, as the seeds. Then, we grow iteratively each of these seeds with neighboring lexes that contribute to a larger cosine similarity, until neighbors no longer contribute.

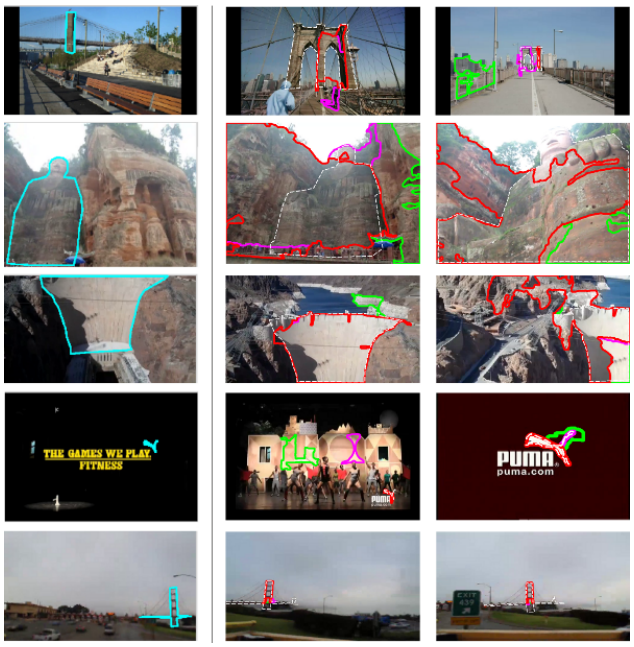


Figure 4. **Segmented object retrieval.** On the left of the gray line we have the query image with the user-specified object of interest. On the right the top two retrieved results, with the top three estimates for the regions of interest. The white striped lines denote the ground truth. Although only a single query example is used, we can identify the segmented object of interest. This is especially noteworthy for the *Brooklyn Bridge* (top row), where the segmented query is viewed sideways and the retrieved segments are photographed from a frontal view and varying distances.

We perform the segmented object retrieval using each image in the dataset, with its given binary mask, as a query. Our evaluation criterion for this experiment is the intersection over union overlap between the segmented query and our top guesses for the top two retrieved images. Similar to [9], we select the top 1 and 3 guesses per retrieved image. When we select only the top one guess, the average overlap is 0.24 and 25% of the images pass the PASCAL criterion that requires at least 50% overlap [11]. When we select the top three guesses the average overlap accuracy increases to 0.27, while 30% of the retrieved images pass the PASCAL criterion. We plot the top three guesses for our five random query images in Figure 4. Although no object classifier is available at query time, codemaps find satisfactory segments in the retrieved images using a single query image only.

7. Conclusions

In this paper, we propose codemaps to segment, classify and search objects locally. Codemaps reorder the encoding, pooling and classification steps of object classification. They do so by exploiting the homomorphic properties of the sum operator and grouping of local neighborhood scores over lattice elements. Our first contribution is introduction of codemaps with ℓ_2 normalization for arbitrarily shaped image regions. Depending on the number of regions analyzed the normalized codemaps are up to 56x faster than traditional Fisher vectors. The fast normalization enables us to reach a better than state-of-the-art performance in semantic segmentation [5] by inclusion of Fisher codemaps. Our second contribution is the embedding of nonlinearities in the codemap decomposition by local kernel pooling. When using the same lattice across images, it allows us to incorporate proven effective explicit and approximate feature mappings [32]. The contribution brings us on par with the state-of-the-art in object classification [23], but 3x faster. Finally, we demonstrate that the efficiency gains of codemaps facilitate object segment retrieval from a single query image. Besides segmentation, classification and search, we anticipate that other computer vision challenges may profit from codemaps as well.

Acknowledgments This research is supported by the STW STORY project, the Dutch national program COMMIT, and by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20067. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

[1] P. Arbelaez, B. Hariharan, C. Gu, S. Gupta, L. D. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *CVPR*, 2012.

[2] P. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, 2009.

[3] L. Bo and C. Sminchisescu. Efficient match kernels between sets of features for visual recognition. In *NIPS*, 2009.

[4] Y.-L. Boureau, J. Ponce, and Y. Lecun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 2010.

[5] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic Segmentation with Second-Order Pooling. In *ECCV*, 2012.

[6] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *TPAMI*, 2002.

[7] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV SLCV*, 2004.

[8] G. Csurka and F. Perronnin. An efficient approach to semantic segmentation. *IJCV*, 2011.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[10] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2012. www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[11] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[12] K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *JMLR*, 2007.

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

[14] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *TPAMI*, 2012.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[16] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr. Graph cut based inference with co-occurrence statistics. In *ECCV*, 2010.

[17] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *TPAMI*, 2009.

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

[19] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *TPAMI*, 2009.

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[21] S. Maji and A. Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009.

[22] D. B. P. Yadollahpour and G. Shakhnarovich. Discriminative Re-Ranking of Diverse Segmentations. In *CVPR*, 2013.

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

[24] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, 2003.

[25] B. Schölkopf and A. J. Smola. *Learning with kernels*. the MIT Press, 2002.

[26] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[27] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR*, 2006.

[28] J. R. R. Uijlings, A. W. M. Smeulders, and R. J. H. Scha. The visual extent of an object. *IJCV*, 2012.

[29] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. In *IJCV*, 2013.

[30] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *TPAMI*, 2010.

[31] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. Visual word ambiguity. *TPAMI*, 2010.

[32] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *TPAMI*, 2011.

[33] A. Vezhnevets, V. Ferrari, and J. M. Buhmann. Weakly supervised structured output learning for semantic segmentation. In *CVPR*, 2012.

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

[35] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[36] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 2007.