Nested Sparse Quantization for Efficient Feature Coding

Xavier Boix[1,⋆], Gemma Roig[1,⋆⋆],
Christian Leistner[1], and Luc Van Gool[1,2]

[1] Computer Vision Lab, ETH Zurich, Switzerland
[2] KU Leuven, Belgium
{boxavier,gemmar,vangool}@vision.ee.ethz.ch

Abstract. Many state-of-the-art methods in object recognition extract features from an image and encode them, followed by a pooling step and classification. Within this processing pipeline, often the encoding step is the bottleneck, for both computational efficiency and performance. We present a novel assignment-based encoding formulation. It allows for the fusion of assignment-based encoding and sparse coding into one formulation. We also use this to design a new, very efficient, encoding. At the heart of our formulation lies a quantization into a set of k-sparse vectors, which we denote as sparse quantization. We design the new encoding as two nested, sparse quantizations. Its efficiency stems from leveraging bitwise representations. In a series of experiments on standard recognition benchmarks, namely Caltech 101, PASCAL VOC 07 and ImageNet, we demonstrate that our method achieves results that are competitive with the state-of-the-art, and requires orders of magnitude less time and memory. Our method is able to encode one million images using 4 CPUs in a single day, while maintaining a good performance.

1 Introduction

Recent research has led to important progress in visual object recognition and classification. Many state-of-the-art object recognition schemes consist of a feed-forward architecture, usually divided into feature extraction, feature encoding, spatial pooling, and a classifier [1,2,3]. Recently, sophisticated encoding and pooling schemes have allowed for learning the different blocks in the pipeline [4,5].

Feature encoding is one of the most intriguing blocks in a feed-forward architecture. Encoding aims at partitioning the feature descriptor space into informative regions, in order to both generalize towards intra-class variances and discriminate between different categories. Many authors pointed out the significant influence feature encoding has on object recognition performance, and stressed the need for encodings with better generalization properties [6,7]. The trend has been to introduce refined encodings that certainly generalize better and thus

⋆ Both first authors are the main contributors, and contributed equally.
⋆⋆ Corresponding author.

© Springer-Verlag Berlin Heidelberg 2012

achieve higher levels of performance, but that come at the cost of increasing computational complexity. Indeed, feature encoding can be orders of magnitude more demanding than feature extraction and pooling, and it can become the bottleneck of the whole pipeline. For real-time and large scale applications, it is therefore crucial to design efficient feature encoding methods.

In the literature, one finds a plethora of feature encoding techniques. One of the most popular is assignment-based coding (AC). AC encodes a feature by assigning it to one or more codebook entries. When combined with average pooling, this corresponds to Bag-of-Words (BoW) [8,9]. Although simple in principle, it can be computationally demanding, since it requires the calculation of distances between the descriptors and the codebook entries. In particular, its computational cost increases with the size of the codebook and the number of patches [10]. Therefore, several methods were devised to speed up the BoW approach, e.g. [11,12,13].

Sparse coding has emerged as a powerful alternative to AC, showing better results than AC, especially, when combined with max-pooling [2,14,15,16]. In terms of efficiency, however, sparse coding optimizes a convex problem of the size of the codebook length, which makes it comparably slow. Locality-constrained linear encoding (LLC) [17] is an approximation of sparse coding, which speeds it up without a significant loss in performance. The current state-of-the-art in visual classification makes use of super-vector coding [18] and Fisher encoding [19], *cf.* [3]. Yet, both methods are memory-demanding. Using compression methods, *e.g.* [19], the Fisher kernel can be made tractable for large-scale applications, but the need for decompression before classification reduces the efficiency again.

In this paper, we focus on AC because it is a promising compromise in terms of speed and performance [6]. We introduce a new formulation for AC, and from that, we design a new efficient feature encoding scheme. At the heart of our formulation lies a quantization into a set of k-sparse vectors, which we denote as *sparse quantization*. It offers a novel viewpoint of AC, which, although algorithmically equivalent to AC, it allows for the unification of AC and sparse coding. In fact, our formulation allows for the design of a new efficient encoding. We coin it 'Nested Sparse Quantization' (NSQ), which consists of two feature encodings, in a nested architecture. We first encode the features, and the result is then fed to another feature encoder. Both encoders follow the novel formulation of AC. The first one is instantiated in a way that it is very fast to execute, and since it is build from a hard assignment encoder, it yields a binary vector. The second encoder is fed with this binary vector and thus also becomes very efficient to compute.

In the experimental part, we compare NSQ to state-of-the-art methods for object recognition. As we show on various datasets, *i.e.*, Caltech 101, PASCAL VOC 07 and ImageNet, our method demands orders of magnitude less time and memory than those previous approaches, while achieving competitive levels of performance. For instance, it is able to encode one million images in less than 24 hours using one laptop of 4 CPUs and 6GB of disk.

2 Preliminaries

In this section, we introduce the mathematical tools that we use in the rest of the paper, *i.e.*, quantization, sparsity and sparse quantization. All proofs can be found in the Supplementary Material.

Quantization. Let $\mathbf{x} \in \mathbb{R}^q$ be a vector, and $\mathbf{B} \in \mathbb{R}^{q\times m}$ the so called codebook matrix with m entries $\mathbf{b}_i \in \mathbb{R}^q$, *i.e.*, $\mathbf{B} = [\mathbf{b}_1 \dots \mathbf{b}_m]$.

Definition 1. Quantization *is a mapping of a vector* $\mathbf{x}$ *into its closest vector of the codebook* $\{\mathbf{b}_i | i = 1, \dots, m\}$*, which we denote as* $\hat{\mathbf{x}}^\star = \arg\min_{\hat{\mathbf{x}}\in\{\mathbf{b}_i\}} \|\hat{\mathbf{x}} - \mathbf{x}\|^2$.

The purpose of quantization is to reduce the cardinality of the representation space. $\mathbf{x}$ has an infinite set of possible values, and when mapped to the codebook, it is restricted to a finite set of possible vectors. In general, the cost of computing a quantization is $\mathcal{O}(m\jmath)$, where $\jmath$ represents the cost of computing a single distance. In practice, the most common used is the squared Euclidean distance, which is expensive since it implies $2q$ float operations, and yields a total cost of $\mathcal{O}(mq)$.

Sparsity. A vector $\boldsymbol{\alpha}$ is k-sparse when it has at most k non-zero entries, *i.e.*, $\|\mathbf{x}\|_0 \leq k$. Let $\mathbb{R}^m_k$ be the space of k-sparse vectors in $\mathbb{R}^m$, which is $\mathbb{R}^m_k = \{\boldsymbol{\alpha} \in \mathbb{R}^m : \|\boldsymbol{\alpha}\|_0 \leq k\}$. Sparsity is a highly non-linear model [20]. Observe that the the sum of two k-sparse vectors does not necessarily result in another k-sparse vector, since their non-zero entries might not coincide. Thus, the result is no longer in $\mathbb{R}^m_k$. We also introduce the subset of binary vectors in $\mathbb{R}^m_k$, which is the set of vectors with k ones and $(m-k)$ zeros. Let $\{0,1\}^m_k = \mathbb{B}^m_k$ be the space of *binary* k-sparse vectors, $\mathbb{B}^m_k = \{\boldsymbol{\alpha} \in \mathbb{B}^m : \|\boldsymbol{\alpha}\|_0 = k\}$, where $\mathbb{B}^m_k$ is a subset of $\mathbb{R}^m_k$, *i.e.*, $\mathbb{B}^m_k \subset \mathbb{R}^m_k$, and it is composed of a finite amount of elements, in particular, its cardinality is $|\mathbb{B}^m_k| = \binom{m}{k}$.

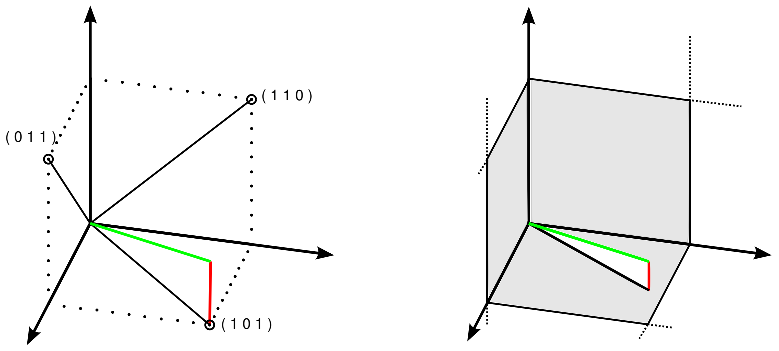


Fig. 1. Representation of $\mathbb{B}^3_2$ (left) and $\mathbb{R}^3_2$ (right) space with the input vector $\mathbf{x}$ in green, and the codification error vector $(\hat{\mathbf{x}} - \mathbf{x})$ in red

Sparse Quantization. We consider the particular case of the quantization when the codebook is the set of k-sparse vectors.

Definition 2. Sparse Quantization *is a quantization into the codebook* $\mathbf{B} = \mathbb{R}^q_k$.

The formulation for the sparse quantization of $\mathbf{x} \in \mathbb{R}^q$ is

$$\hat{\mathbf{x}}^\star = \arg \min_{\hat{\mathbf{x}} \in \mathbb{R}^q_k} \|\hat{\mathbf{x}} - \mathbf{x}\|^2. \tag{1}$$

For brevity, we use the term *Sparse Quantization* (SQ), but to be more precise, it should be called quantization into the codebook $\mathbb{R}^q_k$. Interestingly, sparse quantization can be done in a more efficient way than a quantization into an arbitrary codebook. In the following Proposition, we show that SQ can be achieved with a sorting algorithm, and it does not require the computation of the distances to the codebook entries. The Proposition shows only the case for the codebook $\mathbb{B}^q_k$, which is what we use in our approach. The derivation for sparse quantization in the case where the codebook is the set $\mathbb{R}^q_k$ can be found in the Supplementary Material. We assume that $\mathbf{x} \in \mathbb{R}^q_+$, and that $\mathbf{x}$ is normalized to 1. In all cases where we use the Proposition this is true.

Proposition 1. *Let* $\hat{\mathbf{x}}^\star = \arg\min_{\hat{\mathbf{x}} \in \mathbb{B}^q_k} \|\hat{\mathbf{x}} - \mathbf{x}\|^2$ *be the quantization into* $\mathbb{B}^q_k$ *of* $\mathbf{x} \in \mathbb{R}^q_+$, $\|\mathbf{x}\|_2^2 = 1$. *We can obtain* $\hat{\mathbf{x}}^\star$ *by*

$$\hat{x}^\star_i = \begin{cases} 1 \textit{ if } i \in k\textit{-Highest}(\mathbf{x}) \\ 0 \textit{ otherwise} \end{cases}, \tag{2}$$

where k*-Highest*$(\mathbf{x})$ *is the set of dimensions indices that indicate which are the* k *highest values in the vector* $\mathbf{x}$.

Proposition 1 shows that the sparse quantization can be done with a sorting algorithm of the set $\{x_i\}$, and selecting the k highest values. It has a computational cost of $\mathcal{O}(q)$, in contrast to a general quantization, which has a cost of $\mathcal{O}(mq)$, where in practice $m \gg q$. In Figure 1, we show an example of sparse quantization in $\mathbb{B}^3_2$ and in $\mathbb{R}^3_2$. We plot an input vector $\mathbf{x}$ in green, and the quantization error in red. Also, we show all $\binom{3}{2} = 3$ possible vectors for $\mathbb{B}^3_2$.

3 A New Formulation of Assignment-Based Coding

In this section, we introduce a new framework for Assignment-based Coding (AC), which is based on the principles of sparse quantization. Additionally, we show the relation to Sparse Coding.

3.1 Assignment-Based Coding Revisited

We first review AC and its main variants and identify some common terminology in the literature. This will serve as the basis for our new formulation that we introduce in the subsequent section.

Let $\mathbf{x} \in \mathbb{R}_{+}^{q}$ be the vector of a patch descriptor, which has been normalized, and $\mathbf{B} \in \mathbb{R}^{q \times m}$ the codebook matrix with m entries $\mathbf{b}_i \in \mathbb{R}^q$. AC aims at encoding $\mathbf{x}$ by selecting the codebook entries $\mathbf{b}_i$ with minimum distance to $\mathbf{x}$. Such codebook entries are called the k-Nearest Neighbors (k-NN) of $\mathbf{x}$. k is the number of selected codebook entries, and it is a predefined constant.

Vector quantization (VQ) is the simplest AC method. It only selects a single codebook entry, which is the closest to $\mathbf{x}$, with the so called 1-NN. It is in fact a quantization, since it maps $\mathbf{x}$ into a codebook entry. Let $\boldsymbol{\alpha} \in \mathbb{R}^m$ be the vector that encodes the assignment. In VQ, α_i is set to 1 when it is the element of the codebook that is used to quantize $\mathbf{x}$, and the rest is set to 0.

VQ has been extended to the popular hard-assignment (HA), which assigns more than one codebook entry to $\mathbf{x}$, *e.g.* [12]. In this case, there are 1s placed in the elements of $\boldsymbol{\alpha}$ that correspond to the k-NN codebook entries, *i.e.*,

$$\alpha_i = \begin{cases} 1 \text{ if } i \in k\text{-NN}(\mathbf{x}, \mathbf{B}) \\ 0 \text{ otherwise} \end{cases} . \tag{3}$$

k-NN$(\mathbf{x}, \mathbf{B})$ is the set that indicates which k codebook entries are closest to $\mathbf{x}$. According to Definition 1, HA is not a quantization anymore but an assignment, because more than one vector of the codebook is assigned to $\mathbf{x}$. In the case of quantization, $\mathbf{x}$ is always represented with only a single codebook vector.

In HA, $\boldsymbol{\alpha}$ results in a binary vector with k ones and $(m-k)$ zeros. Thus, the $\boldsymbol{\alpha}$'s obtained are k-sparse vectors, which are in $\mathbb{B}_k^m$. In particular, $\boldsymbol{\alpha}$ is in $\mathbb{B}_k^m$ when using HA, and in $\mathbb{B}_1^m$ for VQ.

There exists also the soft version of HA, denoted as soft-assignment (SA) [6][1]. In this case, instead of indicating the k-NN with 1s, $\boldsymbol{\alpha}$ contains some notion of the relative distance to the selected codebook entries, and becomes

$$\alpha_i = \begin{cases} \exp(-\beta d(\mathbf{x}, \mathbf{b}_i)) & \text{if } i \in k\text{-NN}(\mathbf{x}, \mathbf{B}) \\ 0 & \text{otherwise} \end{cases} . \tag{4}$$

$d(\mathbf{x}, \mathbf{y})$ is a given distance function, β a learned constant, and $d(\mathbf{x}, \mathbf{b}_i)$ is the distance between $\mathbf{x}$ and $\mathbf{b}_i$. Thus, SA codifies a patch descriptor $\mathbf{x}$ with a k-sparse $\boldsymbol{\alpha}$ in $\mathbb{R}_k^m$. Note that when $\beta = 0$ we recover HA, and $\boldsymbol{\alpha} \in \mathbb{B}_k^m$.

AC and other codings are usually the computational bottleneck of the patch-based image classification, *cf.* [3]. Typically, the cost of coding scales with the number of visual words, the number of patch descriptors and the dimensionality of these descriptors. This can make the coding an order of magnitude slower than the patch description. The reason is mainly because the encoding requires computing the distances between the patch descriptor and all the codebook entries.

In the following sections, we introduce a new formulation for AC, that also leads up to a very efficient quantization scheme.

[1] In [6], $\boldsymbol{\alpha}$ is normalized such that $\|\boldsymbol{\alpha}\|_2^2 = 1$. In this paper, for the sake of simplicity in the encoding block, we assume it is the pooling stage that normalizes $\boldsymbol{\alpha}$. Note also that in this paper we refer to SA for what [6] calls Localized SA.

3.2 AC as a Sparse Quantization

We now introduce a new view of AC based on a non-linear mapping of $\mathbf{x}$, which we denote as $\phi : \mathbb{R}^q \to \mathbb{R}^m$.

Definition 3. *Let $\phi(\mathbf{x})$ be the following mapping:*

$$\phi(\mathbf{x}) = \frac{1}{Z}[\exp(-\beta d(\mathbf{x}, \mathbf{b}_1)) \ \ldots \ \exp(-\beta d(\mathbf{x}, \mathbf{b}_m))], \tag{5}$$

where β is a learned constant and $d(a, b)$ a given distance. Z normalizes the vector, and $\{\mathbf{b}\}$ parametrize the mapping.

Observe that in AC, $\mathbf{B}$ denotes the codebook matrix, but in our new formulation it is used to parametrize the mapping $\phi(\mathbf{x})$. In the following Proposition, we rewrite HA as a *sparse quantization* of $\phi(\mathbf{x})$ (section 2). We show this fact for SA as well as the proof in the Supplementary Material.

Proposition 2. *Let $\boldsymbol{\alpha}^\star \in \mathbb{B}_k^m$ be the result of Hard Assignment in* (3)*. Then, the same $\boldsymbol{\alpha}^\star$ can be obtained through the following sparse quantization of $\phi(\mathbf{x})$:*

$$\boldsymbol{\alpha}^\star = \arg \min_{\boldsymbol{\alpha} \in \mathbb{B}_k^m} ||\boldsymbol{\alpha} - \phi(\mathbf{x})||_2^2. \tag{6}$$

Proposition 2 shows that HA is a sparse quantization of $\phi(\mathbf{x})$ into the codebook $\mathbb{B}_k^m$. HA is formulated as a non-linear transformation $\phi(\mathbf{x})$, which uses $\mathbf{B}$, and then the resulting vector is quantized into $\mathbb{B}_k^m$. This is in contrast to how HA has been usually interpreted. Previous HA formulations used $\boldsymbol{\alpha}$ in order to indicate to which codebook entries $\mathbf{x}$ is assigned. Both formulations obtain the same coding results, and have the same computational complexity. However, in the next subsection we show that Proposition 2 is able to fuse Assignment-based Coding and Sparse Coding into a single formulation. Also, we show that we can use the new formulation to design a very efficient codification.

3.3 A Formulation for Sparse Coding and AC

When considering AC as a sparse quantization, we can analyze its relation to sparse coding. Sparse coding is the following coding scheme [21]: $\boldsymbol{\alpha}^\star = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}_k^m} ||\mathbf{C}\boldsymbol{\alpha} - \mathbf{x}||_2^2$, where $\mathbf{C} \in \mathbb{R}^{q \times m}$ is the codebook used in sparse coding. It is usually relaxed in the following way $\boldsymbol{\alpha}^\star = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^m} ||\mathbf{C}\boldsymbol{\alpha} - \mathbf{x}||_2^2 + \lambda\|\boldsymbol{\alpha}\|_1$, where $\lambda\|\boldsymbol{\alpha}\|_1$ is the convex sparsity regularization term of $\boldsymbol{\alpha} \in \mathbb{R}_k^m$, and it can be tuned to enforce different degrees of sparsity. Note that in [15] it was already shown that Sparse Coding can be seen as an extension of VQ; however, their formulation does not extend to AC in general.

The general formulation for Sparse Coding and Assignment-based Codings is:

$$\boldsymbol{\alpha}^\star = \arg \min_{\boldsymbol{\alpha} \in \mathbb{R}_k^m} ||\mathbf{C}\boldsymbol{\alpha} - \phi(\mathbf{x})||_2^2, \tag{7}$$

where $\phi(\mathbf{x})$ represents a mapping from the original q-dimensional vector $\mathbf{x}$ to a space with a possibly different dimension, such as m in the earlier definition. We now recover Sparse Coding by just setting $\phi(\mathbf{x}) = \mathbf{x}$, and we recover AC by setting $\mathbf{C}$ to be the identity matrix $\mathbf{I}^{m \times m}$.

4 Nested Sparse Quantization

In this section, we introduce an efficient encoding built upon the new formulation of HA in Proposition 2. It consists of two sparse quantizations placed in a nested architecture, coined nested sparse quantization (NSQ). We define NSQ as the following optimization problem:

$$\boldsymbol{\alpha} = \arg\min_{\boldsymbol{\alpha}\in\mathbb{B}_k^m} \|\boldsymbol{\alpha} - \phi(\hat{\mathbf{x}})\|^2, \tag{8}$$

$$\text{where } \hat{\mathbf{x}} = \arg\min_{\hat{\mathbf{x}}\in\mathbb{B}_{k'}^{m'}} \|\hat{\mathbf{x}} - \theta(\mathbf{x})\|^2, \tag{9}$$

where (8) is the *outer* sparse quantization and (9) is the *inner* sparse quantization, respectively. The parameters in the inner quantization are m', k' and the mapping $\theta(\mathbf{x})$, and they are not necessarily the same as in the outer quantization. The inner quantization results in the binary $\hat{\mathbf{x}}$, as a result of the sparse quantization into $\mathbb{B}_k^m$. The mapping in the inner quantization $\theta(\mathbf{x})$ has the real-valued $\mathbf{x}$ as input, whereas the outer mapping $\phi(\hat{\mathbf{x}})$ gets the vector $\hat{\mathbf{x}} \in \mathbb{B}_{k'}^{m'}$ as input, which is a binary vector.

4.1 Implementation Advantages

Observe that the input to the outer quantization has a *finite* amount of possible values, because it receives one of the entries of the inner codebook. This allows to use a look up table for the outer quantization, by memorizing the quantization of all entries of the inner codebook. However, in practice, the cardinality of the inner codebook is huge. In NSQ, the outer quantization is a mapping from a set of $\binom{m'}{k'}$ elements to $\binom{m}{k}$ possible outputs, where in practice $\binom{m'}{k'} \gg \binom{m}{k}$ and they are too high to be memorized in a look up table.

Indeed, the advantage of NSQ results from an appropriately chosen nesting of two different quantizations. The inner quantization turns real-valued vectors $\mathbf{x}$ into one of the $\binom{m'}{k'}$ possible binary vectors $\hat{\mathbf{x}}$, and the non-linear mapping should be simple not to overload the system. The outer quantization starts from a vector in the set $\mathbb{B}_{k'}^{m'}$, which are binary vectors, and can therefore be used to very efficiently deploy a more complex mapping $\phi(\hat{\mathbf{x}})$.

Inner Quantization. For the inner quantization, we use the most efficient and simplistic mapping possible, which is $\theta(\mathbf{x}) = \mathbf{x}$; that is, the mapping $\theta(\mathbf{x})$ does not apply any transformation on $\mathbf{x}$. Thus, the inner quantization in Eq. (9) directly quantizes the input $\mathbf{x}$ into the k'-sparse codebook for that step, without any previous non-linear mapping. It becomes

$$\hat{\mathbf{x}} = \arg\min_{\hat{\mathbf{x}}\in\mathbb{B}_{k'}^{q}} \|\hat{\mathbf{x}} - \mathbf{x}\|^2, \tag{10}$$

where m' is equal to q, and k' is the only parameter to be set. As stated in Proposition 1, optimizing this problem is equivalent to computing the k'-Highest

Algorithm 1. Nested Sparse Quantization

```
**Input**: $\mathbf{x} \in \mathbb{R}^q_+$, $\hat{\mathbf{B}} \in \mathbb{B}^{q \times m}_{k'}$
**Output**: $\boldsymbol{\alpha}$
$\hat{\mathbf{x}} \leftarrow$ Set $k'$ highest values of $\mathbf{x}$ to 1 and the rest to 0
**foreach** $i = \{1, \dots, m\}$ **do**
    $\phi_i(\hat{\mathbf{x}}) = \sum_{j=1}^{q} (\hat{x}_j \odot \hat{b}_{ij})$
**end**
$\boldsymbol{\alpha} \leftarrow$ Set $k$ highest values of $\phi(\hat{\mathbf{x}})$ to 1 and the rest to 0
```

values in $\{x_i\}$. This implies selecting the k' elements of $\mathbf{x}$ with the highest values, which has a computational cost of $\mathcal{O}(q)$. One could investigate alternatives to $\theta(\mathbf{x}) = \mathbf{x}$ to improve on the classification results, but this would reduce the speed again and is therefore not studied in this paper.

Outer Quantization. The outer quantization now gets the binary output of the inner quantization as its input, which is a vector in $\mathbb{B}^{m'}_{k'}$. The use of binary vectors allows to reduce the complexity by using fast bit-wise comparisons, which most modern CPUs handle with dedicated instructions [22,23]. Thus, we can afford applying a more sophisticated mapping ϕ. Recall that the computational cost of such $\phi(\hat{\mathbf{x}})$ still is $\mathcal{O}(m\jmath)$, with $\jmath$ the cost of computing the distance. We can reduce the cost $\jmath$ by the very fact that $\hat{\mathbf{x}}$ is binary. Thus, the mapping is the same as in (5) but changing the distance function to a Hamming distance, $\hat{d}(\hat{\mathbf{x}}, \hat{\mathbf{b}}_i)$. Recall that the $\hat{\mathbf{b}}_i$ are parameters used in the definition of the non-linear mapping, and do not correspond neither to the codebook vectors of the inner quantization nor to the codebook vectors of the outer quantization. In our case, these are obtained by piping a total of m vectors $\mathbf{b}_i$ through the inner quantization, thus yielding those $\hat{\mathbf{b}}_i$.

Next, the mapping $\phi(\hat{\mathbf{x}})$ can be further simplified by dropping the exponential functions, given the equivalence of $\arg\min_{\boldsymbol{\alpha}}$ with the mapping $\phi(\hat{\mathbf{x}}) = \frac{1}{Z}[\tilde{d}(\hat{\mathbf{x}}, \hat{\mathbf{b}}_1) \ \dots \ \tilde{d}(\hat{\mathbf{x}}, \hat{\mathbf{b}}_m)]$. Now, suppose we take $\tilde{d}(\hat{\mathbf{x}}, \hat{\mathbf{b}}_i) = \sum_{j=1}^{q} (\hat{x}_j \odot \hat{b}_{ij})$, where $\odot$ is the negation of the exclusive OR operator (*nxor*). It returns 1 if both elements are equal, and 0 otherwise. This is again very efficient to compute.

In Algorithm 1 we depict the implementation of NSQ, which highlights the simplicity and efficiency of the method.

4.2 Discussion

Quantization vs. recognition accuracy At this point the reader may object that NSQ adds an extra quantization step in the feature encoding procedure, and that quantization errors may grow worse, leading to a deteriorated performance. Yet, in recognition, quantization should also allow for sufficient generalization. In the next section, we show that recognition performance is not impaired, but that the speed goes up dramatically.

Relation of the Inner Quantization to Binary Embeddings and Hashing. NSQ is not to be confused with Binary Embeddings or Hashing-based methods. The goal of the inner quantization is a first step of feature encoding, and it is not designed to be a fast nearest neighbor extractor nor a fast distance approximator. In fact, in the experiments we show that NSQ is not a good nearest neighbor approximator. Hashing-based methods tackle fast distance computation rather than feature encoding, and have been typically used to speed up image retrieval in large-scale databases [24,25,26,27]. The binarizations used by these methods range from thresholding linear combinations of input features to solving optimization problems. These have a negligible cost in image retrieval, but not when applied to feature encoding. For example, in Locality Sensing Hashing (LSH) the binary embedding is $h_n(x) = T(\mathbf{r}^T\mathbf{x})$, where $\mathbf{r}$ is a projection vector, and $T(\cdot)$ is a threshold function. Hashing uses L different N-dimensional embedding functions $h(x)$ of an image signature $\mathbf{x}$, and thus, the cost of computing $h(\mathbf{x})$ is $\mathcal{O}(LNq)$. Hashing is typically used in applications where the cost $\mathcal{O}(LNq)$ is negligible compared to the cost of searching the nearest-neighbors, because this search might be among hundreds of thousands of elements. However, in feature encoding, the time of computing $h(\mathbf{x})$ is not negligible anymore, because the nearest neighbor search is only among thousands of elements (the size of the codebook), and in addition, $h(x)$ is computed for each patch. In contrast, the inner quantization of NSQ obtains the binary $\hat{\mathbf{x}}$ at a cheap cost $\mathcal{O}(q)$. Furthermore, the binary embeddings generate each bit independently of the others [26], which is different from NSQ, in which the output bits are dependent since only k of them are 1.

5 Experiments

In this section, we report experiments on different datasets, namely Caltech101 [28], PASCAL VOC 2007 [29] and ImageNet [30]. In none of them we use flipped or blurred images to extend the training set.

Caltech 101. It contains 102 different classes with about 50 images per class. We use 3 random splits of 30 images per class for training and the rest for testing, and evaluate it with the average classification accuracy across all classes. We resize the image to have a maximum of 300 pixels per dimension.

PASCAL VOC 2007. It consists of a total of 9,963 images with 20 different object classes. Half of the dataset is used for training and the other half for testing. The evaluation is based on the mean average precision (mAP) across all classes.

ImageNet. We create a new dataset taking a subset of $1,065,687$ images of ImageNet. This subset contains images of 909 different classes that do not overlap in the synset. We randomly split this subset into two halfs, one for training and one for testing, maintaining the proportion of images per class. For evaluation, we report the average classification accuracy across all classes. Note that we do not use the protocol of taking the 5 highest scores per image and keeping the best. We resize the image to have a maximum of 300 pixels per dimension.

5.1 Implementation Details

Patch Descriptors. We use SIFT [31] extracted from patches on a regular grid, at different scales. In Caltech 101 and ImageNet they are extracted at every 8 pixels and at the scales of 16, 32 and 48 pixels diameter. In VOC07, SIFT is sampled at each 4 pixels and at the scales of 12, 24 and 36 pixels diameter.

Feature Encoding. Apart from comparing NSQ and HA, we also report results with LLC [17] and the Super Vector method [18]. We found that for all settings $k = 5$ is the optimal, except for the combination of HA, max-pooling and linear SVM, which is $k = 10$. In NSQ, $k' = 25$ for the inner quantization (see discussion in 5.2 for those choices).

Codebook Generation. The codebook $\mathbf{B}$ is typically built with a learning algorithm either unsupervised [8,9] or supervised [13]. Recently, Coates and Ng [7] showed that a similar performance can be achieved by randomly picking a set of patches as codebook entries. We compared a codebook obtained using k-means clustering versus building it taking random patches from the training set. Tallying with the observations in [7], compared to k-means clustering, random selection considerably reduces the training time but does not lower the performance for large codebooks. For the methods that are not AC, like LLC or Super Vector, we learned the vocabularies as specified for each of them.

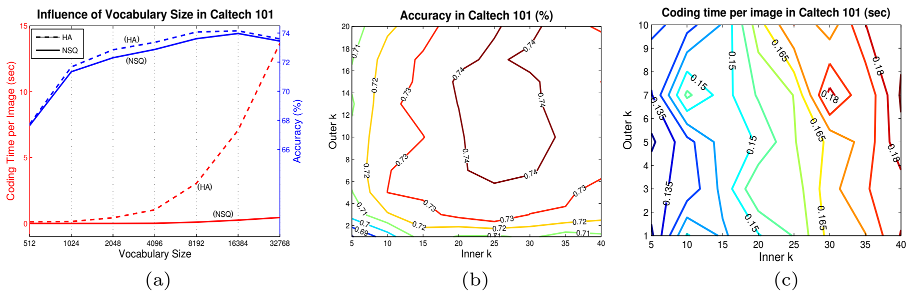


Fig. 2. *Influence of the parameter k and the vocabulary size in Caltech 101.* (a) Comparison of performance and time between NSQ and HA when varying the size of the codebook. (b) Accuracy and (c) Time when varying the inner and outer quantization parameters.

Pooling. We use in all datasets spatial pyramids. In Caltech 101 and ImageNet we divide the image in 4×4, 2×2 and 1×1 regions, yielding 21 different regions, and in VOC07, 3×1, 2×2 and 1×1 regions, yielding 8 regions. We use max-pooling because it has been shown to systematically outperform other types of pooling. In the case of NSQ and HA, the max-pooling results in a binary vector, which is the descriptor of the whole image, which we do not normalize.

Classification. For Caltech 101 and VOC07, we use a linear one versus rest SVM classifier for each class with the parameter C of the SVM set to 1000. In Caltech 101, we also report results using the RB-χ^2 kernel, $\exp\{-\beta\chi^2\}$, setting β to the

Table 1. *Quantitative Results on Caltech 101.* The time and memory measurements are for one image. We refer to $\text{HA}_{\{k',k\}}$ for the HA with NSQ, and $\text{HA}_{\{k\}}$ for the standard, in which we indicate the k and k' parameters of the outer and inner quantizations, respectively.

	Method				**Time per Image 1 CPU** *(sec)*			**Memory**
	Coding	**Pool**	**Kernel**	**Acc.** *(%)*	**Feature**	**Coding**	**Kernel**	*(Bytes)*
NSQ	$\text{HA}_{\{25,10\}}$	Max	Linear	$74.2 \pm .7$	$0.27 \pm .11$	$0.17 \pm .05$	–	21K
	$\text{HA}_{\{20,5\}}$	Max	Linear	$73.7 \pm .3$	$0.27 \pm .11$	$0.15 \pm .05$	–	21K
	$\text{HA}_{\{20,5\}}$	Max	RB-χ^2	$72.0 \pm .6$	$0.27 \pm .11$	$0.15 \pm .05$	.02	21K
	$\text{HA}_{\{20,5\}}$	Sum	RB-χ^2	$71.6 \pm .3$	$0.27 \pm .11$	$0.17 \pm .07$	2.10	672K
Standard	$\text{HA}_{\{10\}}$	Max	Linear	$74.6 \pm .6$	$0.27 \pm .11$	$2.78 \pm .9$	–	21K
	$\text{HA}_{\{5\}}$	Max	Linear	$74.2 \pm .1$	$0.27 \pm .11$	$2.75 \pm .9$	–	21K
	$\text{HA}_{\{5\}}$	Max	RB-χ^2	$73.4 \pm .9$	$0.27 \pm .11$	$2.75 \pm .9$	.02	21K
	$\text{HA}_{\{5\}}$	Sum	RB-χ^2	$72.1 \pm .4$	$0.27 \pm .11$	$2.78 \pm .9$	2.10	672K
	LLC	Max	Linear	$76.1 \pm .7$	$0.27 \pm .11$	$2.95 \pm .9$	–	672K
	SV	Max	Linear	$76.9 \pm .9$	$0.27 \pm .11$	$1.52 \pm .9$	–	10752K
Other features	HA [6]	Max	Linear	$73.3 \pm .6$	–	–	–	–
	SA [6]	Max	Linear	$74.2 \pm .8$	–	–	–	–
	LLC [17]	Max	Linear	$73.4 \pm -$	–	–	–	–
	FK [3]	Max	Linear	$77.8 \pm .6$	–	–	–	–

mean of the pairwise distances in the training data. Before computing the χ^2 distance, the features are normalized with the ℓ_1-norm. When the image descriptor is binary, the RB-χ^2 kernel can be computed efficiently using dedicated CPU instructions [22,23]. In ImageNet we use as classifier an approximated nearest neighbor that takes a maximum of 400 random examples per class.

Computational Evaluation. We use CPUs at 3.07GHz with the SSE4.1 instruction set, that enables fast popcnt instructions for bit strings.

5.2 Results

Performance Evaluation. In Table 1 we report results in Caltech 101. We show the results for NSQ in combination with different SVM kernels and pooling schemes. We do the same for HA, and we also include a comparison with LLC and Super Vector. We use a codebook of $8,192$ entries for all methods except for Super Vector for which we use $1,024$, since it achieves the compromise between accuracy and efficiency. LLC and Super Vector perform better than the others, as was reported in previous papers. However, their coding consumes much more memory and time than NSQ. The performance of NSQ is only 2.7% less than the best competing method which is Super Vector, but NSQ requires 500 times less memory than Super Vector and is about 10 times faster, and is 20 times faster than other methods that use the same codebook size. In Table 2 we summarize the results in VOC07. We use a codebook size of $16,258$ entries. NSQ obtains a 30 times speed up, while only degrading 2% in performance.

In both tables we also provide the state-of-the-art methods in the literature. Even though they obtain better performance than our method, these techniques come at a far higher computational cost, as seen in the previous comparisons. We did not reproduce results of SA because it was already observed by [6] that

Table 2. *PASCAL VOC 2007 classification results.* The average score provides the per-class average. The time and memory measurements are for one image, using a single CPU.

	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dinning Table	Dog	Horse	Motorbike	Person	Potted Plant	Sheep	Sofa	Train	TV/Monitor	**Average**	**Feature Time**	**Coding Time**	**Memory**
NSQ	67	56	41	66	21	56	72	55	41	48	42	37	71	65	78	21	45	53	71	51	52.87	0.85	0.86	42K
HA	67	56	44	66	22	57	75	56	44	50	43	37	74	69	78	24	46	56	74	51	54.54	0.85	21.22	42K
LLC	69	58	46	70	21	60	74	57	43	52	44	42	73	68	81	24	48	57	72	52	55.56	0.85	23.81	1344K
LLC [17]	75	65	51	71	29	69	79	62	54	49	52	44	77	67	84	31	45	53	79	54	59.30	–	–	–
SV [18]	79	72	55	74	34	72	83	64	57	53	63	49	81	72	85	36	46	60	83	59	64.00	–	–	–

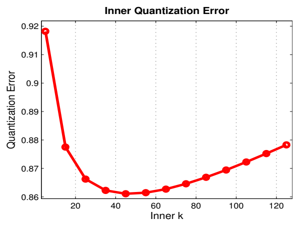
SA only outperforms HA by less than 1%. Also, note that in VOC07 there is a gap between the performance of LCC reported by [17] and the LCC using our descriptors. According to [3], this can be explained by the fact that [17] extend the training set by blurring and flipping the images, and we do not.

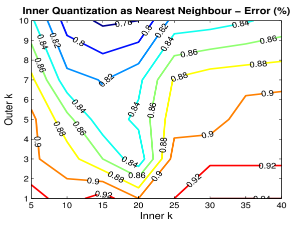
Influence of the Parameters. In Figure 2a, we compare time and accuracy when changing the codebook size for NSQ and HA. Up to a certain point, the larger the codebook, the better the accuracy becomes. Interestingly, the computational time for HA explodes when increasing the codebook size, while in NSQ the increase is negligible compared to HA.

We analyze the accuracy and time performance of our method when changing the quantization parameters. We use the first split of Caltech 101 to show the results. In Figures 2b and 2c, we compare the accuracy of our method when changing the k parameter for the inner quantization and the outer quantization, and keeping the codebook size at $8,192$. We see a peak of performance: When the inner k is around 20 and 35 and the outer k is around 10 and 16. In the right plot we analyze the time when changing this parameters, also keeping the codebook at size $8,192$. The lower both k are, the less time classification takes. The best compromise between time and performance is using $k' = 25$ for the inner quantization, and $k = 10$ for the outer.

Influence of the Quantization Error. We analyze the impact of having more quantization error due to the inner quantization. We use Caltech 101 to show results using $8,192$ codebook entries. In Figure 3a, we show that the quantization error of the inner quantization does not have a clear relation with the classification accuracy. The accuracy is maximal for k' at 20 and 35, and the quantization error has its minimum around 45. Moreover, we show the percentage of equal activated elements of $\boldsymbol{\alpha}$ in NSQ and in HA using the same $\mathbf{B}$, in Figure 3b. Around 90% of the nearest neighbors are different. This shows that NSQ is a new encoding and not an approximate nearest-neighbor.

ImageNet in a Laptop in one Day. We introduce a new benchmark to test the efficiency of the encodings. We used ImageNet with restrictions on time and





	NSQ	**HA**
Time Dataset		
Features	17h 21m	17h 21m
Coding	6h 54m	7h 9m
Total	24h 18m	24h 30m
SizeDataset	5.4GB	700MB
Acc.Top-1	20.01%	16.6%

(a) (b) (c)

Fig. 3. (a) Quantization error of the inner quantization vs k', in Caltech 101. (b) Evaluation of the inner quantization as a nearest neighbor on Caltech 101 changing k and k'. (c) *Quantitative results on ImageNet.* To fulfill the time constraints, NSQ uses a codebook of $2,048$ and HA of 256 .

memory, and only let the image encoding algorithms run about 24 hours on a laptop of 4 CPUs. We compared our method with HA with max pooling in which we adjusted the parameters to fulfill the requirements. With NSQ we were able to use a codebook of $2,048$ entries, and in HA only of 256. In Figure 3b the results are summarized. In this benchmark, we outperform HA because in the same amount of time NSQ can use a larger codebook, which is crucial to have good classification performance. The reported results are of the same order of magnitude as reported in [32], though a direct comparison is not possible because we use a different subset of ImageNet. The state-of-the-art for this dataset is reported by [33], but the computational effort of this method is huge compared to ours.

6 Conclusion

We investigated efficient encoding schemes for object recognition and presented a novel encoding formulation that allows to fuse sparse coding and assignment-based coding. We proposed a very efficient encoding algorithm called Nested Sparse Quantization (NSQ), which is based on two nested quantizations. The implementation of NSQ is very simple, and can virtually be incorporated everywhere where assignment-based coding schemes are employed, but higher efficiency and more compact representations are demanded, *e.g.* real-time and large scale recognition problems. This potential was corroborated by our experiments on the Caltech 101, PASCAL VOC 07 and ImageNet benchmarks.

Acknowledgments. This work was partially supported by the EU projects FP7-ICT-24314 Interactive Urban Robot (IURO) and FP7-ICT-248873 Robotic ADaptation to Humans Adapting to Robots (RADHAR).

References

1. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: CVPR (2008)

2. Boureau, Y., Bach, F., LeCun, Y., Ponce, J.: Learning mid-level features for recognition. In: CVPR (2010)
3. Chatfield, K., Lempitsky, V., Vedaldi, A., Zisserman, A.: The devil is in the details: An evaluation of recent feature encoding methods. In: BMVC (2011)
4. Zeiler, M.D., Taylor, G.W., Fergus, R.: Adaptive deconvolutional networks for mid and high level feature learning. In: ICCV (2011)
5. Yu, K., Lin, Y., Lafferty, J.: Learning image representations from pixel level via hierarchical sparse coding. In: CVPR (2011)
6. Liu, L., Wang, L., Liu, X.: In defence of soft-assignment coding. In: ICCV (2011)
7. Coates, A., Ng, A.: The importance of encoding versus training with sparse coding and vector quantization. In: ICML (2011)
8. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: ECCV (2004)
9. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV (2003)
10. Nowak, E., Jurie, F., Triggs, B.: Sampling Strategies for Bag-of-Features Image Classification. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 490–503. Springer, Heidelberg (2006)
11. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: CVPR (2006)
12. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: Improving particular object retrieval in large scale image databases. In: CVPR (2008)
13. Moosmann, F., Jurie, F., Triggs, B.: Fast discriminative visual codebooks using randomized clustering forests. In: NIPS (2007)
14. Boureau, Y., Ponce, J., LeCun, Y.: A theoretical analysis of feature pooling in vision algorithms. In: NIPS (2010)
15. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR (2009)
16. Benoit, L., Mairal, J., Bach, F., Ponce, J.: Sparse image representation with epitomes. In: CVPR (2011)
17. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR (2010)
18. Zhou, X., Yu, K., Zhang, T., Huang, T.S.: Image Classification Using Super-Vector Coding of Local Image Descriptors. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 141–154. Springer, Heidelberg (2010)
19. Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher Kernel for Large-Scale Image Classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 143–156. Springer, Heidelberg (2010)
20. DeVore, R.: Nonlinear approximation. Acta Numerica (1998)
21. Olshausen, B., Field, D.J.: Sparse coding with an overcomplete basis set: A strategy employed by v1? Vis. Res. (1997)
22. Shakhnarovich, G.: Learning Task-Specic Similarity. PhD thesis, Massachusetts Institute of Technology (2005)
23. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: Binary Robust Independent Elementary Features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 778–792. Springer, Heidelberg (2010)
24. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: STOC (1998)

25. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS (2008)
26. Gordo, A., Perronnin, F.: Asymmetric distances for binary embeddings. In: CVPR (2011)
27. Jégou, H., Matthijs Douze, C.S.: Product quantization for nearest neighbor search. PAMI (2011)
28. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. PAMI (2006)
29. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge (VOC 2007) (2007) Results
30. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR 2009 (2009)
31. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV (2004)
32. Deng, J., Berg, A.C., Li, K., Fei-Fei, L.: What Does Classifying More Than 10,000 Image Categories Tell Us? In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 71–84. Springer, Heidelberg (2010)
33. Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K.: Large-scale image classification: fast feature extraction and svm training. In: CVPR (2011)