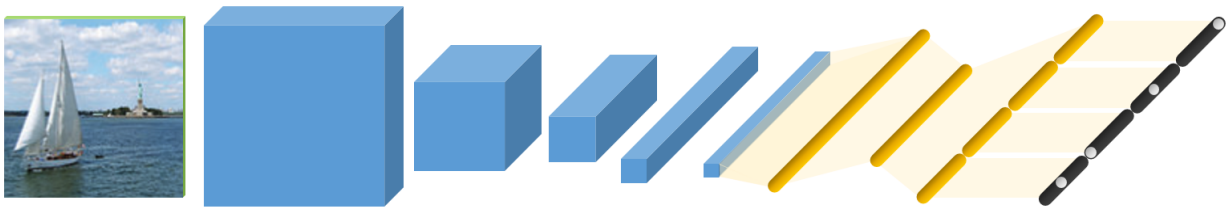
SuBiC: A Supervised, Structured Binary Code for Image Search

Himalaya Jain, Joaquin Zepeda, Patrick Perez, Rémi Gribonval

SUBIC: A supervised, structured binary code for image search

Himalaya Jain[1,2], Joaquin Zepeda[3], Patrick Pérez[1], and Rémi Gribonval[2]

[1]Technicolor, Rennes, France [2]INRIA Rennes, France [3]Amazon, Seattle, WA



We introduce SUBIC, a *supervised structured binary code* (concatenation of one-hot blocks). It is produced by a novel supervised deep convolutional network and is well adapted to efficient visual search, including category retrieval for which it outperforms the state-of-the-art supervised deep binary hashing techniques.

Abstract

For large-scale visual search, highly compressed yet meaningful representations of images are essential. Structured vector quantizers based on product quantization and its variants are usually employed to achieve such compression while minimizing the loss of accuracy. Yet, unlike binary hashing schemes, these unsupervised methods have not yet benefited from the supervision, end-to-end learning and novel architectures ushered in by the deep learning revolution. We hence propose herein a novel method to make deep convolutional neural networks produce supervised, compact, structured binary codes for visual search. Our method makes use of a novel block-softmax nonlinearity and of batch-based entropy losses that together induce structure in the learned encodings. We show that our method outperforms state-of-the-art compact representations based on deep hashing or structured quantization in single and cross-domain category retrieval, instance retrieval and classification. We make our code and models publicly available online.

1. Introduction

Deep convolutional neural networks (CNNs) have proven to be versatile image representation tools with great generalization power, a quality that has rendered them indispensable in image search. A given network trained on the ImageNet dataset [16], for example, can achieve excellent performance when transferred to a variety of other datasets [20, 30], or even to other visual search tasks [41]. This quality of *transferability* is important in large-scale image search, where the time or resources to compile annotations in order to train a new network for every new dataset or task are not available.

A second desirable property of image representations for large-scale visual search is that of being *compact yet functional*. A paramount example of such a representation is provided by image indexing schemes, such as Product Quantization (PQ) [24] and others, that rely on vector quantization with structured, unsupervised codebooks [2, 12, 47]. PQ can be seen as mapping a feature vector into a binary vector consisting of a concatenation of one-hot encoded codeword indices. One can directly compare an uncompressed query feature with these binary vectors by means of an inner product between the binary vectors and a real-valued mapping of the query feature vector.

It is not surprising that, with the dawn of the deep learning revolution, many recent research efforts have been directed towards supervised learning of deep networks that produce compact and functional binary features [15, 17, 31, 32, 33, 45, 46, 48]. One commonality between these approaches – which we refer to collectively as *deep hashing* methods – is their reliance on element-wise binarization mechanisms consisting of either sigmoid/tanh nonlinearities [15, 31, 32, 33, 45, 46, 48] or element-wise binarizing penalties such as [15, 17]. Indeed, to our knowledge, ours is the first approach to impose a structure on the learned binary representation: We employ two entropy-based losses

to induce a one-hot block structure in the produced binary feature vectors, while favoring statistical uniformity in the support of the active bits of each block. The resulting structured binary code has the same structure as a PQ-encoded feature vector.

Imposing structure on the support of the binary representation has two main motivations: First, structuring allows a better exploitation of the binary representation's support to encode semantic information, as exemplified by approaches that learn to encode face parts [5], visual attributes [36] and text topics [26] in the support of the representation under weak supervision. We promote this desirable property by means of an entropy-based loss that encourages uniformity in the position of the active bits – a property that would not be achievable using a simple softmax non-linearity. Note that, as a related added benefit, the structuring makes it possible to use a binary representation of larger size without incurring extra storage. Second, the structuring helps in regularizing the architecture, further contributing to increased performance relative to other, non-structured approaches.

While all previously existing deep hashing methods indeed produce very compact, functional representations, they have not been tested for transferability. The main task addressed in all these works is that of category retrieval wherein a given test example is used to rank all the test images in all classes. Yet all deep hashing approaches employ a *single-domain* approach wherein the test classes and training classes are the same. It has been established experimentally [42] that excellent performance can be achieved in this test by simply assigning to each stored database image, the class label produced by a classifier trained on the corresponding training set. Hence, it is also important to test for *cross-domain* category retrieval, wherein the architecture learned on a given set of training classes is tested on a new, disjoint set of test classes. We present experiments of both types in this work, outperforming several baselines in the cross-domain test and recent deep hashing methods in the single-domain test.

The contributions of the present work can be summarized as follows:

- We introduce a simple, trainable, CNN layer that encodes images into structured binary codes that we coin SUBIC. While all other approaches to supervised binary encoding use element-wise binarizing operations and losses, ours are block-based.

- We define two block-wise losses based on code entropy that can be combined with a standard classification loss to train CNNs with a SUBIC layer.

- We demonstrate that the proposed binary features outperform the state-of-the art in single-domain category retrieval, two competitive baselines in cross-domain category retrieval and image classification, and state-of-the art unsupervised quantizers in image retrieval.

- Our approach enables asymmetric search with a search complexity comparable to that of deep hashing.

2. Related work

We discuss here the forms of vector quantization and binary hashing that are the most important for efficient visual search with compact codes, and we explain how our approach relates to them.

Unsupervised structured quantization. Vector quantization (VQ), *e.g.* with unsupervised k-means, is a classic technique to index multi-dimensional data collections in a compact way while allowing efficient (approximate) search. Structured versions of VQ, *e.g.* product, additive or composite [2, 18, 19, 23, 24, 28, 38, 47], have established impressive indexing systems for large scale image collections. Coupled with single or multiple index inverted file systems [1, 25], these VQ techniques currently offer state-of-the-art performance for very large-scale high-dimensional nearest-neighbor search (relative to the Euclidean distance in input feature space) and instance image search based on visual similarity. All these unsupervised quantization techniques operate on engineered or pre-trained image features. Owing to the success of CNNs for image analysis at large, most recent variants use off-the-shelf or specific CNN features as input representation, *e.g.*, [3, 27, 34, 44]. However, contrary to binary hashing methods discussed below, VQ-based indexing has not yet been approached from a supervised angle where available semantic knowledge would help optimize the indexed codes and possibly the input features. In the present work, we take a supervised encoding approach that bears a strong connection to supervised binary hashing, while exploiting an important aspect of these powerful unsupervised VQ techniques, namely the structure of the code. The binary codes produced by our approach are in a discrete product space of size K^M while allowing $\mathcal{O}(M \log K)$ storage and $\mathcal{O}(M)$ search complexity.

Deep, supervised hashing. Binary hashing is a long-standing alternative to the above-mentioned VQ methods, and the deep learning revolution has pushed the state-of-the-art of these approaches. Deep supervised hashing methods – CNNH+ [45], DRSCH [46], DSRH [48], DNNH [31], DLBHC [32], DSH [33] and BDNN [17] – share the following high-level principles. An off-the-shelf or home-brewed convolutional network f_1 is used to extract a high-dimensional distributed representation $\mathbf{x} \in \mathbb{R}^d$ from an input image $\mathbf{I}$. A subsequent fully connected encoding layer f_2 turns this feature vector into a compact binary code

Table 1: Comparison of proposed approach to recent supervised binary hashing techniques.

Method	supervision	binarization (train – test)	code-based loss on	base CNN training	cross-domain
CNNH+ [45]	pair-wise	sigmoid–threshold	dist. to target code	yes	no
DRSCH [46]	triplet-wise	sigmoid–threshold	none	yes	no
DSRH [48]	triplet-wise	sigmoid–threshold	training average	yes	no
DNNH [31]	triplet-wise	sigmoid–threshold	none	yes	no
DLBHC [32]	point-wise	sigmoid–threshold	none	fine-tuning	no
DSH [33]	pair-wise	sigmoid–threshold	distance to binary	yes	no
BDNN [17]	pair-wise	built-in	feature reconst. error	no	no
SUBIC (ours)	point-wise	block softmax–argmax	block-based entropies	yes	yes

$\mathbf{h} \in \{0,1\}^B$ of B bits through final entry-wise thresholding (or sign function for centered codes), B typically ranging from 12 to 64 bits. At training time, this binarization is usually relaxed using a sigmoid (or $\tanh$ for centered codes) – with the exception of BDNN [17] –, which results in an encoding layer that outputs vectors in $\in [0,1]^B$. Using semantic supervision, f_2 is trained while f_1 is fixed to pre-trained values, fine-tuned or trained from scratch. Supervision coming from class labels is used either directly (classification training) [32] or using tuples (pairs [17, 33, 45] or triplets [31, 46, 48]) as in metric learning. Table 1 summarizes the specifics of each of these methods.

DLBHC [32] is a simple instance employing a sigmoid-activated encoding layer grafted to the pre-trained AlexNet architecture [29] and trained using a standard classification objective. Other methods employ additional training loss(es) at the code level to induce desirable properties. BDNN [17] uses the code-to-feature reconstruction error, making the approach applicable in an unsupervised regime. DSH [33] employs a W-shaped loss with minima at the desired code values, while DSRH [48] penalizes the average of each bit over the training set such that its distribution is approximately centred (final code is in $\{-1,+1\}^B$). CNN+ [45] proposes the direct supervision of hash functions with target binary codes learned in a preliminary phase via low-rank factorization of a full pairwise similarity matrix. Note also that DRSCH [46] learns bit-wise weights along with the binary encoder, which results in richer codes but more costly distances to compute at search time.

As described in detail next, our approach follows the same high-level principles discussed above, but with important differences. The first, key difference lies in the structure of the codes. We define them as the concatenation of M one-hot vectors (binary vectors with all entries but one being zero) of size K. This gives access to K^M distinct codes, hence corresponding to an effective bit-size of $M \log_2 K$ bits, as in VQ methods that combine M codewords, each taken from a codebook of size K. Binarization and its relaxed training version thus operate at the block level. This specific code structure is combined with novel loss terms that enforce respectively the one-sparsity of each block and the effective use of the entire block support. Also, contrary to most supervised binary hashing approaches, with the exception of [32], we resort only to point-wise supervision.

To our knowledge, only one other approach has incorporated product-wise structuring within a deep learning pipeline [7]. Yet that method does not learn the structuring as part of a deep architecture, relying rather on a standard product quantizer that is updated once per epoch in an unsupervised manner.

3. Approach

We describe in this section the design of our SUBIC architecture, its supervised training and its use for visual search.

3.1. Architecture

Following the approach discussed above, we consider the following classification feed-forward network (Fig. 1):

$$\mathbf{s} \triangleq FC_1 \circ f_2 \circ f_1(\mathbf{I}), \tag{1}$$

where $\mathbf{I}$ is an input image, f_1 a deep CNN with L convolutional layers (inc. pooling and normalization, if any) and Q fully-connected layers, f_2 a binary encoding layer, FC_1 a C-class classification layer, and $\mathbf{s}$ the C-dimensional vector of class-probability estimates.

We aim for the binary encoding layer f_2 to produce structured binary vectors $\mathbf{b}$ consisting of the concatenation of M one-hot encoded vectors $\mathbf{b}_m, m = 1,\dots,M$, of dimension K, *i.e.*, $\mathbf{b} = [\mathbf{b}_1;\dots;\mathbf{b}_M]$.[1] Formally, the *blocks* $\mathbf{b}_m$ should satisfy

$$\mathbf{b}_m \in \mathcal{K}_K \triangleq \{\mathbf{d} \in \{0,1\}^K \text{ s.t. } \|\mathbf{d}\|_1 = 1\}. \tag{2}$$

Accordingly, our codes $\mathbf{b}$ should come from the discrete product set $\mathcal{K}_K^M$.

In practice, f_2 employs a fully-connected layer FC_0 with ReLU non-linearity producing real-valued vectors $\mathbf{z} \in \mathbb{R}_+^{KM}$ likewise consisting of M K-dimensional blocks $\mathbf{z}_m$.

[1] Using vector stacking notation $[\mathbf{a};\mathbf{b}] = [\mathbf{a}^\top, \mathbf{b}^\top]^\top$, where $\mathbf{a}$ and $\mathbf{b}$ are column-vectors.

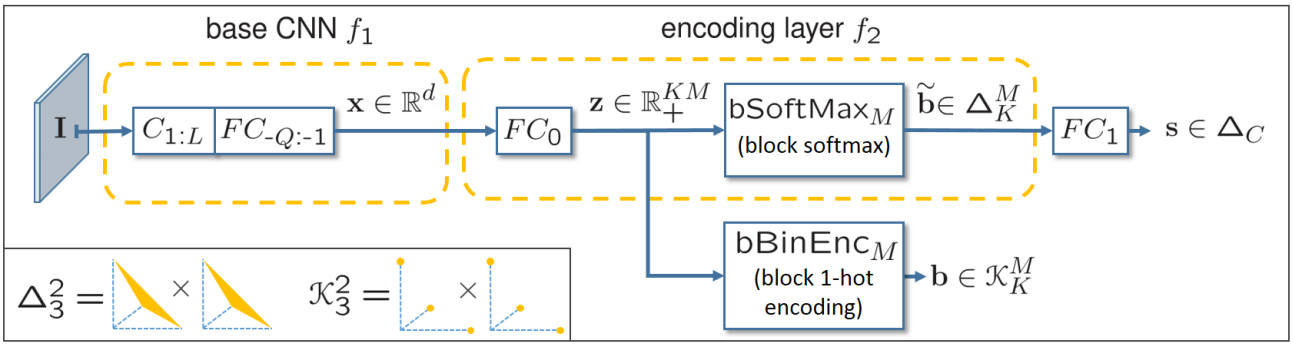


Figure 1: **Proposed architecture and notations**. A feature is extracted from image $\mathbf{I}$ by a base CNN f_1 and binarized using a block-structured encoding layer f_2 consisting of a fully-connected layer followed by a *block softmax* during training, or a *block* 1*-hot encoder* during testing.

A second non-linearity operates on each $\mathbf{z}_m$ to produce the corresponding binarized block. We use a different binarization strategy at training time (top branch in Fig. 1) and at test time (bottom branch), as discussed next.

Training architecture. Similarly to supervised binary hashing approaches discussed in Section 2, we enable back-propagation during our learning process by relaxing the structured binarization constraint (2), producing instead structured real-valued codes $\widetilde{\mathbf{b}} \in \Delta_K^M$, where

$$\Delta_K \triangleq \{\mathbf{d} \in [0,1]^K \text{ s.t. } \|\mathbf{d}\|_1 = 1\} \tag{3}$$

is the convex hull of $\mathcal{K}_K$ (see Fig. 1 bottom-left, for the examples Δ_3^2 and $\mathcal{K}_3^2$). We achieve this by introducing the *block-softmax non-linearity* $\widetilde{\mathbf{b}} = \text{bSoftMax}_M(\mathbf{z})$ (*c.f.* Fig. 1) which computes the blocks $\widetilde{\mathbf{b}}_m$ from the corresponding blocks $\mathbf{z}_m$ as follows ($\exp(\cdot)$ denotes element-wise exponentiation):

$$\widetilde{\mathbf{b}}_m = \frac{1}{\|\exp(\mathbf{z}_m)\|_1} \exp(\mathbf{z}_m). \tag{4}$$

Test time architecture. At test time, the block softmax is replaced (*cf.* Fig. 1, bottom branch) by a *block one-hot encoder* $\mathbf{b} = \text{bBinEnc}_M(\mathbf{z})$, which uses $\mathbf{z}$ to efficiently compute the projection of each block $\widetilde{\mathbf{b}}_m \in \Delta_K$ onto $\mathcal{K}_K$ using

$$\mathbf{b}_m[k] = \begin{cases} 1 & \text{if } k = \text{argmax}_r\, \mathbf{z}_m[r], \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $\mathbf{d}[k]$ denotes the k-th entry of a vector $\mathbf{d}$. Note, particularly, that $\text{bBinEnc}_M(\mathbf{z}) = \text{bBinEnc}_M(\widetilde{\mathbf{b}})$.

3.2. Supervised loss and training

In order to bring real-valued code vectors $\widetilde{\mathbf{b}}$ as close as possible to block-wise one-hot vectors, while making the best use of coding budget, we introduce two entropy-based losses that will be part of our learning objective. Our approach assumes a standard learning method wherein training examples $(\mathbf{I}^{(i)}, y^{(i)})$ consisting of an image $\mathbf{I}^{(i)}$ and its class label $y^{(i)} \in \{1, \ldots, C\}$ are divided into mini-batches $\{(\mathbf{I}^{(i)}, y^{(i)})\}_{i \in \mathcal{T}}$ of size $|\mathcal{T}| = T$.

Our losses will be based on *entropy*, which is computed for a vector $\mathbf{p} \in \Delta_K$ as follows:[2]

$$\text{E}(\mathbf{p}) \triangleq -\sum_{k=1}^{K} \mathbf{p}[k] \log_2 \mathbf{p}[k]. \tag{6}$$

Entropy is smooth and convex and further has the interesting property that it is the theoretical minimum average number of bits per symbol required to encode an infinite sequence of symbols with distribution $\mathbf{p}$ [14]. Accordingly, it is exactly zero, its minimum, if $\mathbf{p}$ specifies a deterministic distribution (*i.e.*, $\mathbf{p} \in \mathcal{K}_K$) and $\log_2 K$, its maximum, if it specifies a uniform distribution (*i.e.*, $\mathbf{p} = \frac{1}{K}\mathbf{1}$).

Toward block-wise one-hot encoding. Given the merits of structured binary codes discussed previously, we aim to produce feature vectors $\widetilde{\mathbf{b}} = [\widetilde{\mathbf{b}}_1; \ldots; \widetilde{\mathbf{b}}_M]$ consisting of blocks $\widetilde{\mathbf{b}}_m$ that approximate one-hot encoded vectors, thus that have a small projection error

$$\min_{\mathbf{d} \in \mathcal{K}_K} \|\mathbf{d} - \widetilde{\mathbf{b}}_m\|_2. \tag{7}$$

In the ideal case where $\widetilde{\mathbf{b}}_m \in \mathcal{K}_K$, it has minimum entropy of 0. The convexity/smoothness of $\text{E}(\cdot)$ means that blocks with low entropy will have small projection error (7), thus suggesting penalizing our learning objective for a given training image using $\sum_m \text{E}(\widetilde{\mathbf{b}}_m)$. We overload our definition of $\text{E}(\cdot)$ in (6) and let $\text{E}(\widetilde{\mathbf{b}}) \triangleq \sum_m \text{E}(\widetilde{\mathbf{b}}_m) \in$

[2] With the usual convention $0 \log_2(0) = 0$.

$[0, M \log_2 K]$. Accordingly, we refer to the average of these losses over a training batch $\mathcal{T}$ as the *mean entropy*, given by

$$\frac{1}{TM} \sum_{i \in \mathcal{T}} \sum_{m=1}^{M} \mathrm{E}(\widetilde{\mathbf{b}}_m^{(i)}) = \frac{1}{TM} \sum_{i \in \mathcal{T}} \mathrm{E}(\widetilde{\mathbf{b}}^{(i)}). \quad (8)$$

In practice, introducing this loss will result in vectors $\widetilde{\mathbf{b}}$ that are only approximately binary, and hence, at test time, we project each block $\widetilde{\mathbf{b}}_m$ onto $\mathcal{K}_K$ using (5).

Uniform block support. Besides having blocks $\widetilde{\mathbf{b}}_m$ that resemble one-hot vectors, we would like for the supports of the binarized version $\mathbf{b}_m$ of $\widetilde{\mathbf{b}}_m$ to be as close to uniformly distributed as possible. This property allows the system to better exploit the support of our $\widetilde{\mathbf{b}}$ in encoding semantic information. It further contributes to the regularization of the model and encourages a better use of the available bit-rate.

We note first that one can estimate the distribution of the support of the $\mathbf{b}_m$ from a batch $\mathcal{T}$ using $\frac{1}{T} \sum_{i \in \mathcal{T}} \mathbf{b}_m^{(i)}$. Relaxing $\mathbf{b}_m$ to $\widetilde{\mathbf{b}}_m$ for training purposes, we want the entropy of this quantity to be high. This leads us to the definition of the negative *batch entropy* loss:

$$-\frac{1}{M} \sum_{m=1}^{M} \mathrm{E}\left(\frac{1}{T} \sum_{i \in \mathcal{T}} \widetilde{\mathbf{b}}_m^{(i)}\right) = -\frac{1}{M} \mathrm{E}\left(\overline{\mathbf{b}}\right), \quad (9)$$

where we let $\overline{\mathbf{b}} \triangleq \frac{1}{T} \sum_{i \in \mathcal{T}} \widetilde{\mathbf{b}}^{(i)}$.

Our learning objective (computed over a mini-batch) will hence be a standard classification objective further penalized by the mean and batch entropies in (8) and (9):

$$\mathrm{Loss}\big(\{(\mathbf{I}^{(i)}, y^{(i)})\}_{i \in \mathcal{T}}\big) \triangleq \frac{1}{T} \sum_{i \in \mathcal{T}} \Big[\ell(\mathbf{s}^{(i)}, y^{(i)}) + \frac{\gamma}{M \log_2 K} \mathrm{E}(\widetilde{\mathbf{b}}^{(i)}) - \frac{\mu}{M \log_2 K} \mathrm{E}(\overline{\mathbf{b}})\Big], \quad (10)$$

with network output $\mathbf{s}$ defined as in (1), C the number of classes, and $\gamma > 0$ and $\mu > 0$ two hyper-parameters.

In our work, we use the following scaled version of the commonly used cross-entropy loss for classification:

$$\ell(\mathbf{s}, y) \triangleq -\frac{1}{\log_2 C} \log_2 \mathbf{s}[y]. \quad (11)$$

The scaling by $\log_2 C$ reduces the dependence of the hyper-parameters μ and γ on the number of classes C.

The training loss (10) is minimized with mini-batch stochastic gradient descent. The whole architecture can be learned this way, including the CNN feature extractor, the encoding layer and the classification layer (Fig. 1). Alternatively, (some of) the weights of the base CNN f_1 can be fixed to pre-trained values. In Section 4, we will consider the following variants, depending on set-ups: Training of FC_0/FC_1 only ("2-layer" training), the base CNN staying fixed; Training of $FC_{-1}/FC_0/FC_1$ ("3-layer training"); Training of all layers, $C_1 \cdots C_L$ and $FC_{-Q} \cdots FC_1$ ("full training").

3.3. Image search

As we will establish in Section 4, SUBIC yields important advantages in three image search applications, which we now describe along with a search complexity analysis.

Category and instance retrieval. These two tasks consist of ranking database images according to their similarity to a given query image, where similarity is defined by membership in a given semantic category *(category retrieval)* or by the presence of a specific object or scene *(instance retrieval)*. For these two tasks, we wish to use our structured binary representations to efficiently compute similarity scores for all database of images $\{\mathbf{I}^{(j)}\}_j$ given a query image $\mathbf{I}^*$. We propose using an asymmetric approach [24] that limits query-side coding approximation: The database images are represented using their structured binary representation $\mathbf{b}^{(j)} = [\mathbf{b}_1^{(j)}; \cdots; \mathbf{b}_M^{(j)}] \in \mathcal{K}_K^M$, whereas the query image $\mathbf{I}^*$ is represented using the real-valued vector $\mathbf{z}^* = [\mathbf{z}_1^*; \cdots; \mathbf{z}_M^*] \in \mathbb{R}_+^{KM}$. Accordingly, the database images are ranked using the similarity score $(\mathbf{z}^*)^\top \mathbf{b}^{(j)}$. This expression also reads

$$\sum_{m=1}^{M} \mathbf{z}_m^* \big[\operatorname*{argmax}_r \mathbf{b}_m^{(j)}[r]\big], \quad (12)$$

which shows that M additions are needed to compute SUBIC similarities.

Image classification. A second important application is that of image classification in the case where the classes of interest are not known beforehand or change across time, as is the case of on-the-fly image classification from text queries [11, 10]. Having feature representations that are compact yet discriminative is important in this scenario, and a common approach to achieve this is to compress the feature vectors using PQ [8, 10]. The approach we propose is to instead use our supervised features to compactly represent the database images directly. New classes are assumed to be provided in the form of annotated sets $\{(\mathbf{I}^{(q)}, y^{(q)})\}_q$ containing examples of the C' previously-unknown query classes,[3] and we learn classifiers from the structured codes $\widetilde{\mathbf{b}}$ of these examples. At test time, classifying a test feature $\mathbf{b}^{(j)} \in \mathcal{K}_K^M$ from the original dataset will require computing products $(\boldsymbol{W}^*)^\top \mathbf{b}^{(j)}$ (with $\boldsymbol{W}^* \in \mathbb{R}^{KM \times C'}$ for a softmax

[3] Obtained from an external image search engine in on-the-fly scenarios.

classifier or $\boldsymbol{W}^* \in \mathbb{R}^{KM}$ for a one-vs-rest classifier). Similarly to (12), this operation will likewise require only M additions per column of $\boldsymbol{W}^*$.

Search complexity relative to deep hashing. The expression (12) is reminiscent of the efficient distance computation mechanisms based on look-up-tables commonly used in product quantization search methods [24]. In particular, the expression in (12) establishes that computing the similarity between $\mathbf{z}^*_m$ and $\mathbf{b}_m$ incurs a complexity of M additions. This can be compared to the complexity incurred when computing the Hamming distance between two deep hash codes (*cf.* Section 2) $\mathbf{h}_1$ and $\mathbf{h}_2$ of length $B = M \log_2 K$ (*i.e.*, of storage footprint B equal to that of SUBIC): 1 XOR operation followed by as many additions as there are different bits in $\mathbf{h}_1$ and $\mathbf{h}_2$, a value that can be estimated from the expectation ($[\![\cdot]\!]$ is the Iverson bracket)

$$\mathbb{E}_{\mathbf{h}_1,\mathbf{h}_2}\left(\sum_{k=1}^{B}[\![\mathbf{h}_1[k] \neq \mathbf{h}_2[k]]\!]\right) = \frac{B}{2} = \frac{M}{2}\log_2 K, \tag{13}$$

if assuming i.i.d. and uniform $\mathbf{h}_j[k]$.

We note that $\mathcal{O}(1)$ look-up-table (LUT) based implementations of the Hamming distance are indeed possible, but only for small B (the required LUT size is 2^B). Alternatively, a smaller LUT of size $2^{B/M'}$ can be used by splitting the code into M' blocks (with M' comparable to M), resulting in a complexity $\mathcal{O}(M')$ comparable to the $\mathcal{O}(M)$ complexity of SUBIC.

4. Experiments

We assess the merits of the proposed supervised structured binary encoding for instance and semantic image retrieval by example and for database image classification, the three tasks described in Section 3.3.

Single-domain category retrieval. Single-domain category retrieval is the main experimental benchmark in the supervised binary hashing literature. Following the experimental protocol of [33], we report mean average precision (mAP) performance on the Cifar-10 database [13] which has 10 categories and 60k images of size 32×32 for each. The training is done on the 50k image training set. The test set is split into 9k database images and 1k query images, 100 per class. For fairness of comparison, we also use as base CNN the same as introduced in [33]. It is composed of $L = 3$ convolutional layers with 32, 32 and 64 filters of size 5×5 respectively, followed by a fully connected layer FC_{-1} with $d = 500$ nodes. As proposed, we append to it a randomly initialized *encoder layer* FC_0 along with the classification layer FC_1. We fixed $K = 64$ and varied $M = \{2, 4, 6, 8\}$ so that $B = M \log_2(K)$ is equal to the desired bit-rate. Full training of the network is conducted, and hyper-parameters γ and μ are cross-validated as discussed later. We compare in Table 2 with various methods based on the same base CNN (top four rows, DSH [33], DNNH [31], DLBHC [32] and CNNH+ [45]), as well as other published values. For reference, we include a method (KSH-CNN [35]) not based on neural hash functions but using activations of a deep CNN as input features. Note that, at all bit-rates, from 12 to 48 bits, SUBIC outperforms these state-of-the-art supervised hashing techniques.

Method	12-bit	24-bit	36-bit	48-bit
CNNH+ [45]	0.5425	0.5604	0.5640	0.5574
DLBHC [32]	0.5503	0.5803	0.5778	0.5885
DNNH [31]	0.5708	0.5875	0.5899	0.5904
DSH [33]	0.6157	0.6512	0.6607	0.6755
KSH-CNN [35]	-	0.4298	-	0.4577
DSRH [48]	-	0.6108	-	0.6177
DRSCH [46]	-	0.6219	-	0.6305
BDNN [17]	-	0.6521	-	0.6653
SUBIC (ours)	**0.6349**	**0.6719**	**0.6823**	**0.6863**

Table 2: **Single-domain category retrieval.** Comparison against published mAP values on Cifar-10 for various supervised deep hashing methods. See the *ImageNet* column of Table 3 for single-domain results on ImageNet.

Method	VOC2007	Caltech-101	ImageNet
PQ [24]	0.4965	0.3089	0.1650
CKM [38]	0.4995	0.3179	0.1737
LSQ [37]	0.4993	0.3372	0.1882
DSH-64 [33]	0.4914	0.2852	0.1665
SUBIC 2-layer	**0.5600**	0.3923	0.2543
SUBIC 3-layer	0.5588	**0.4033**	**0.2810**

Table 3: **Cross-domain category retrieval.** Performance (mAP) using 64-bit encoders across three different datasets using VGG-128 as base feature extractor. For completeness, results on ImageNet validation set (*i.e.* single-domain retrieval) are provided in the third column.

Cross-domain category retrieval. Using VGG-D with 128-D bottleneck (VGG-128) [9] as base CNN ($L = 5$, $Q = 3$ and $d = 128$), setting μ and γ to 1.0, we performed 2-layer and 3-layer learning of our network (see Section 3.2) on ILSVRC-ImageNet [22] training set. Two-layer training is conducted on 5k batches of $T = 200$ images. Three-layer training is initialized by previous one and run for another 5k batches. To evaluate cross-domain performance, we used our trained network to do category retrieval on Pascal VOC2007[43], Caltech-101 [6] and ImageNet validation sets. For each experiment, we used 1000 (2000 for ImageNet) random query images, the rest

Method	Oxford5K	Paris6K
PQ [24]	0.2374	0.3597
LSQ [37]	0.2512	0.3764
DSH-64 [33]	0.2108	0.3287
SUBIC	**0.2626**	**0.4116**

Table 4: **Instance retrieval.** Performance (mAP) comparison using 64-bit codes for all methods.

serving as database. The performance of the two trained SUBIC networks is reported in Table 3 at 64-bit rate ($M = 8$, $K = 256$). They are compared to three unsupervised quantization baselines, PQ [24], Cartesian k-means (CKM) [38] and LSQ [37], operating at 64-bit rate on VGG-128 image features. Further, to compare with supervised deep hashing approaches we implemented DSH [33] with VGG-128 as the base CNN, using their proposed loss and pairwise training.

The impact of the proposed semantic supervision across domain is clearly demonstrated. Comparing unsupervised methods with our "2-layer" trained variant (no tuning of FC_1 is particularly enlightening since they all share exactly the same 128-dimensional input features). Training this representation as well in the "3-layer" version did not prove useful except on the ImageNet validation set. Note that the performance on this set could have been further improved through longer training, but at the expense on reduced transferability.

Instance retrieval. Unsupervised structured quantizers produce compact codes that enjoy state-of-the-art performance in instance retrieval at low memory footprint. Hence, in Table 4 we compare SUBIC to various such quantizers as well as DSH, using 64-bit representations for all methods. We used the *clean train* subset [21] of the Landmarks dataset [4] to train both DSH and a 2-layer SUBIC (the same as in Table 3, but with 60K batches). We report mAP on the Oxford 5K [39] and Paris 6K [40] datasets using their provided query/database split. SUBIC outperforms all methods while DSH performance is weaker to even unsupervised quantizers.

Image classification. In Table 5, we show how the 64-bit SUBIC encoding of VGG-128 features from Table 3 (2-layer variant) outperforms two baseline encoders with the same bit-rate for classification of compressed representations. As done in [10] for on-the-fly classification, the first baseline employs PQ [24] to represent the features compactly, and the second substitutes PQ by the better-performing CKM encoder [38]. Both unsupervised encoders are learned on VGG-128 features from the ImageNet training set. For the test on ImageNet, the two baselines employ the off-the-shelf VGG-128 classification layer

	ImageNet		VOC2007
	Top-1 acc.	*Top-5 acc.*	*mAP*
VGG-128*	53.80	77.32	73.79
PQ 64-bit	39.88	67.22	65.94
CKM 64-bit	41.15	69.66	67.25
SUBIC soft*	50.07	74.11	70.20
SUBIC 64-bit	47.77	72.16	67.86

Table 5: **Classification performance with different compact codes**. The rows marked (*) are non-binary codes. See the text for details.

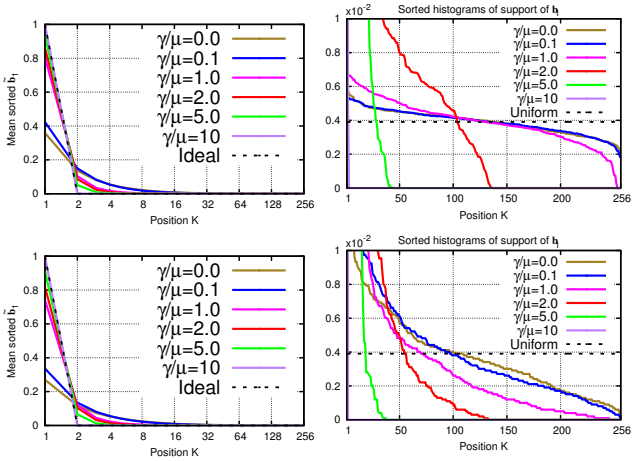


Figure 2: **Effect of the entropy-based losses on the behavior of structured encoding.** (*left*) One-hot encoding closeness of $\widetilde{\mathbf{b}}_1$. (*right*) Distribution of block support of $\mathbf{b}_1$. The black dashed curves correspond to the ideal, desired behavior. (*top*) ImageNet validation. (*bottom*) VOC2007.

as a classifier, reconstructing the PQ and CKM encoded versions beforehand (for reference, first row is for this classifier using original, un-coded features). Our results (bottom two rows) employ trained FC_1 layer applied to either $\widetilde{\mathbf{b}}$ code ("SUBIC soft") or $\mathbf{b}$ binary code ("SUBIC 64-bit"). In case of VOC2007 we trained one-vs-rest SVM classifiers on the off-the-shelf VGG-128 features (top three rows) or on the $\widetilde{\mathbf{b}}$ features for SUBIC (bottom two rows).

Note that our compact SUBIC 64-bit features outperform both PQ and CKM features for the same bit-rate. Also notice that, although the classifiers for VOC2007 are trained on block-softmax encoded features, when we use SUBIC 64-bit features the accuracy drops only marginally.

Structuring effectiveness of entropy-based losses. In Fig. 2 we evaluate our proposed entropy losses using the same SUBIC setup as in Table 5. We report statistics on the ImageNet validation set (top graphs) and on all of VOC2007 (bottom graphs), using the γ/μ ratio in the leg-

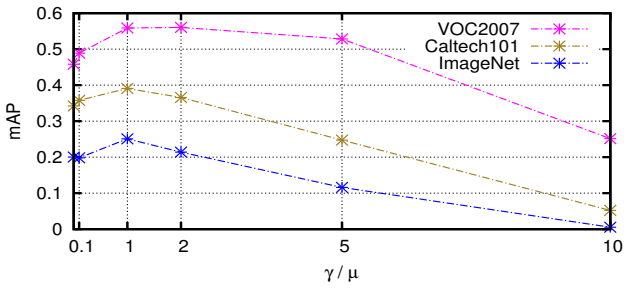


Figure 3: **Effect of γ and μ on category retrieval performance.**

ends for ease of comparison.

To explore how well our γ-weighted mean-entropy loss favors codes resembling one-hot vectors, we extract the first 256-dimensional block $\widetilde{\mathbf{b}}_1$ from each image of the set (the seven other blocks exhibit similar behavior), re-order the entries of each such $\widetilde{\mathbf{b}}_1$ in decreasing order and average the resulting collection of vectors. The entries of this average vector are visualized for various values of γ/μ in the plots on the left. Ideal one-hot behavior corresponds to $[1; 0 \cdots ; 0]^\top$. On both datasets, increasing the mean entropy penalization weight γ relative to μ (*i.e.*, increasing γ/μ) results in code blocks that more closely resemble one-hot vectors.

To evaluate how well our μ-weighted negative batch-entropy term promotes uniformity of the support of the binarized blocks in $\mathbf{b}$, we plot, in the right side of Fig. 2, the sorted histograms of the support of the first block $\mathbf{b}_1$ over the considered image set. Note that increasing the weight μ of the batch entropy term relative to γ (decreasing γ/μ) results in distributions that are closer to uniform. As expected, the effect is more pronounced on the ImageNet dataset (top row) used as a training set, but extrapolates well to an independent dataset (bottom row).

We note further that it is possible (green curves, $\gamma/\mu = 5$) to have blocks that closely resemble one-hot vectors (left plot) but make poor use of the available support (0-valued histogram after the 47-th bin, on the right). It is likewise possible (blue curve, $\gamma/\mu = 0.1$) to enjoy good support usage with blocks that do not resemble one-hot vectors, establishing that our two losses work together to achieve the desired design goals.

Cross-validation of hyper parameters. Using the same setup and γ/μ values as in Fig. 2, in Fig. 3 we plot mAP as a function of γ/μ on three datasets. Note that the optimal performance (at $\gamma/\mu = 1$) for this architecture is obtained for an operating point that makes better use (closer to uniform) of the support of the blocks, as exemplified by the corresponding curves (pink) on the right in Fig. 2. This supports one of our original motivations that fostering uniformity of the support would encourage the system to use the support to encode semantic information.

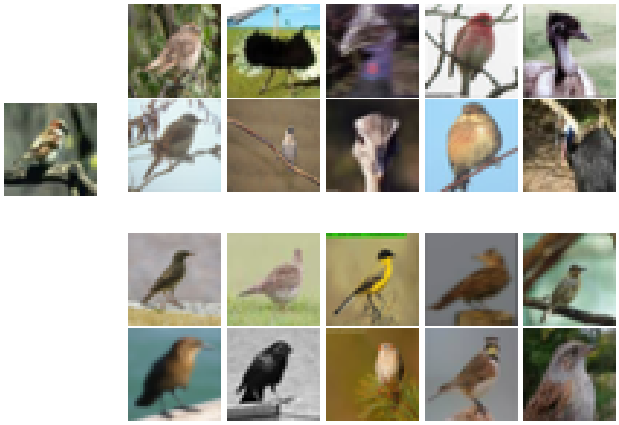


Figure 4: **Category retrieval examples**. Top ten ranked images retrieved from Cifar-10 for the query on the left when using 12-bit (*top*) and 48-bit (*bottom*) SUBIC. Note that higher bit-rates make the representation more sensitive to the query's orientation.

Category retrieval search examples. In Fig. 4 we present a search example when using the 12-bit and 48-bit SUBIC from Table 2. Note that increasing the bit rate results in retrieved images that are of the same pose as the query, suggesting that our method has potential for weakly-supervised (automatic) category refinement.

5. Conclusion

In this work we introduced SUBIC, a supervised, structured binary code produced by a simple encoding layer compatible with recent deep pipelines. Unlike previous deep binary hash codes, SUBIC features are block-structured, with each block containing a single active bit. We learn our proposed features in a supervised manner by means of a block-wise softmax non-linearity along with two entropy-based penalties. These penalties promote the one-hot quality of the blocks, while encouraging the active bits to employ the available support uniformly. While enjoying comparable complexity at fixed bit-rate, SUBIC outperforms the state-of-the art deep hashing methods in the single-domain category retrieval task, as well as state-of-the art structured vector quantizers in the instance retrieval task. SUBIC also outperforms structured vector quantizers in cross-domain category retrieval. Our method further showed promise for weakly-supervised semantic learning, a possible future direction.

References

[1] A. Babenko and V. Lempitsky. The inverted multi-index. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2012. 2

[2] A. Babenko and V. Lempitsky. Additive quantization for extreme vector compression. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2014. 1, 2

[3] A. Babenko and V. Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2016. 2

[4] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *Proc. Europ. Conf. Computer Vision*, 2014. 7

[5] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Structured sparsity through convex optimization. *Statistical Science*, pages 450–468, 2012. 2

[6] Caltech-101dataset. www.vision.caltech.edu/Image_Datasets/Caltech101/. 6

[7] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen. Deep quantization network for efficient image retrieval. In *Proc. AAAI Conf. on Artificial Intelligence*, 2016. 3

[8] K. Chatfield, R. Arandjelović, O. Parkhi, and A. Zisserman. On-the-fly learning for visual search of large-scale image and video datasets. *Int. J. Multimedia Information Retrieval*, 4(2):75–93, 2015. 5

[9] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. British Machine Vision Conf.*, 2014. 6

[10] K. Chatfield, K. Simonyan, and A. Zisserman. Efficient on-the-fly category retrieval using ConvNets and GPUs. In *Proc. Asian Conf. Computer Vision*, 2014. 5, 7

[11] K. Chatfield and A. Zisserman. Visor: Towards on-the-fly large-scale object category retrieval. In *Proc. Asian Conf. Computer Vision*, 2012. 5

[12] Y. Chen, T. Guan, and C. Wang. Approximate nearest neighbor search by residual vector quantization. *Sensors*, 10(12):11259–11273, 2010. 1

[13] Cifar-10 dataset. www.cs.toronto.edu/~kriz/cifar.html. 6

[14] T. M. Cover and J. A. Thomas. *Elements of information theory 2nd edition*. Wiley-interscience, 2006. 4

[15] Q. Dai, J. Li, J. Wang, and Y.-G. Jiang. Binary optimized hashing. In *Proc. ACM Int. Conf. Multimedia*, 2016. 1

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2009. 1

[17] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *Proc. Europ. Conf. Computer Vision*, 2016. 1, 2, 3, 6

[18] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2013. 2

[19] T. Ge, K. He, and J. Sun. Product sparse coding. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2014. 2

[20] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proc. Europ. Conf. Computer Vision*, 2014. 1

[21] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *Proc. Europ. Conf. Computer Vision*, 2016. 7

[22] Ilsvrc-ImageNet dataset. www.image-net.org/challenges/LSVRC/. 6

[23] H. Jain, P. Pérez, R. Gribonval, J. Zepeda, and H. Jégou. Approximate search with quantized sparse representations. In *Proc. Europ. Conf. Computer Vision*, 2016. 2

[24] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Machine Intell.*, 33(1):117–128, 2011. 1, 2, 5, 6, 7

[25] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: Re-rank with source coding. In *Proc. Int. Conf. Acoust., Speech, Signal Processing*, 2011. 2

[26] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal Methods for Sparse Hierarchical Dictionary Learning. In *ICML*, 2010. 2

[27] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *arXiv:1702.08734*, 2017. 2

[28] Y. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2014. 2

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Int. Conf. Neural Inf. Proc. Systems*, 2012. 3

[30] P. Kulkarni, F. Jurie, J. Zepeda, P. Pérez, and L. Chevallier. SPLeaP: Soft pooling of learned parts for image classification. In *Proc. Europ. Conf. Computer Vision*, 2016. 1

[31] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2015. 1, 2, 3, 6

[32] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *Conf. Comp. Vision Pattern Rec. Workshops*, 2015. 1, 2, 3, 6

[33] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2016. 1, 2, 3, 6, 7

[34] R. Liu, Y. Zhao, S. Wei, Z. Zhu, L. Liao, and S. Qiu. Indexing of CNN features for large scale image search. *arXiv preprint arXiv:1508.00217*, 2015. 2

[35] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2012. 6

[36] X. Lyu, J. Zepeda, and P. Pérez. Maximum margin linear classifiers in unions of subspaces. In *Proc. British Machine Vision Conf.*, 2016. 2

[37] J. Martinez, J. Clement, H. H. Hoos, and J. J. Little. Revisiting additive quantization. In *Proc. Europ. Conf. Computer Vision*, 2016. 6, 7

[38] M. Norouzi and D. Fleet. Cartesian k-means. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2013. 2, 6, 7

[39] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2007. 7

[40] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2008. 7

[41] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *Conf. Comp. Vision Pattern Rec. Workshops*, 2014. 1

[42] A. Sablayrolles, M. Douze, H. Jégou, and N. Usunier. How should we evaluate supervised hashing? *arXiv preprint arXiv:1609.06753*, 2016. 2

[43] Pascal VOC2007 dataset. host.robots.ox.ac.uk/pascal/VOC/voc2007/. 6

[44] D. Wang, C. Otto, and A. K. Jain. Face search at scale. *IEEE Trans. Pattern Anal. Machine Intell.*, 2016. 2

[45] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *Proc. AAAI Conf. on Artificial Intelligence*, 2014. 1, 2, 3, 6

[46] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Trans. Image Processing*, 24(12):4766–4779, 2015. 1, 2, 3, 6

[47] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *Proc. Int. Conf. Machine Learning*, 2014. 1, 2

[48] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Proc. Conf. Comp. Vision Pattern Rec.*, 2015. 1, 2, 3, 6