Learning a Complete Image Indexing Pipeline
Himalaya Jain, Joaquin Zepeda, Patrick Perez, Rémi Gribonval

To cite this version:
Himalaya Jain, Joaquin Zepeda, Patrick Perez, Rémi Gribonval. Learning a Complete Image Indexing Pipeline. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun 2018, Salt Lake City, France. pp.4933-4941, 10.1109/CVPR.2018.00518. hal-04517974

HAL Id: hal-04517974
https://inria.hal.science/hal-04517974v1
Submitted on 22 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
Learning a Complete Image Indexing Pipeline

Himalaya Jain¹,², Joaquin Zepeda³, Patrick Pérez¹, and Rémi Gribonval²

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

Abstract

To work at scale, a complete image indexing system comprises two components: An inverted file index to restrict the actual search to only a subset that should contain most of the items relevant to the query; An approximate distance computation mechanism to rapidly scan these lists. While supervised deep learning has recently enabled improvements to the latter, the former continues to be based on unsupervised clustering in the literature. In this work, we propose a first system that learns both components within a unifying neural framework of structured binary encoding.

1. Introduction

Decades of research have produced powerful means to extract features from images, effectively casting the visual comparison problem into one of distance computations in abstract spaces. Whether engineered or trained using convolutional deep networks, such vector representations are at the core of all content-based visual search engines. This applies particularly to example-based image retrieval systems where a query image is used to scan a database for images that are similar to the query in some way: in that they are the same image but one has been edited (near duplicate detection), or because they are images of the same object or scene (instance retrieval), or because they depict objects or scenes from the same semantic class (category retrieval).

Deploying such a visual search system requires conducting nearest neighbour search in a high-dimensional feature space. Both the dimension of this space and the size of the database can be very large, which imposes severe constraints if the system is to be practical in terms of storage (memory footprint of database items) and of computation (search complexity). Exhaustive exact search must be replaced by approximate, non-exhaustive search. To this end, two main complementary methods have emerged, both relying on variants of unsupervised vector quantization (VQ). The first such method, introduced by Sivic and Zisserman [25] is the inverted file system. Inverted files rely on a partitioning of the feature space into a set of mutually exclusive bins. Searching in a database thus amounts to first assigning the query image to one or several such bins, and then ranking the resulting shortlist of images associated to these bins using the Euclidean distance (or some other distance or similarity measure) in feature space.

The second method, introduced by Jegou et al. [15], consists of using efficient approximate distance computations as part of the ranking process. This is enabled by feature encoders producing compact representations of the feature vectors that further do not need to be decompressed when computing the approximate distances. This type of approaches, which can be seen as employing block-structured binary representations, superseded the (unstructured) binary hashing schemes that dominated approximate search.

Despite its impressive impact on the design of image representations [11, 1, 10, 23], supervised deep learning is still limited in what concerns the approximate search system itself. Most recent efforts focus on supervised deep binary hashing schemes, as discussed in the next section. As an exception, the work of Jain et al. [13] employs a block-structured approach inspired by the successful compact encoders referenced above. Yet the binning mechanisms that enable the usage of inverted files, and hence large-scale search, have so far been neglected.

In this work we introduce a novel supervised inverted file system along with a supervised, block-structured encoder that together specify a complete, supervised, image indexing pipeline. Our design is inspired by the two methods of successful indexing pipelines described above, while borrowing ideas from [13] to implement this philosophy.

Our main contributions are as follows: (1) We propose the first, to our knowledge, image indexing system to reap the benefits of deep learning for both data partitioning and feature encoding; (2) Our data partitioning scheme, in particular, is the first to replace unsupervised VQ by a supervised approach; (3) We take steps towards learning the feature encoder and inverted file binning mechanism simultaneously as part of the same learning objective; (4) We establish a wide margin of improvement over the existing
baselines employing state-of-the-art deep features, feature encoders and binning mechanism.

2. Background

Approximating distances through compact encoding

Concerning approximate distance computations, two main approaches exist. Hashing methods [26], on the one hand, employ Hamming distances between binary hash codes. Originally unsupervised, these methods have recently benefited from progress in deep learning [27, 28, 30, 17, 18, 19, 7], leading to better systems for category retrieval in particular. Structured variants of VQ, on the other hand, produce fine-grain approximations of the high-dimensional features themselves through very compact codes [3, 8, 9, 12, 15, 16, 20, 29] that enable look-up table-based efficient distance computations. Contrary to recent hashing methods, VQ-based approaches have not benefited from supervision so far. However, Jain et al. [13] recently proposed a supervised deep learning approach that leverages the advantages of structured compact encoding and yields state-of-the-art results on several retrieval tasks. Our work extends this supervised approach towards a complete indexing pipeline, that is, a system that also includes an inverted file index.

Scanning shorter lists with inverted indexes

For further efficiency, approximate search is further restricted to a well chosen fraction of the database. This pruning is carried out by means of an Inverted File (IVF), which relies on a partitioning of the feature space into Voronoi cells defined using $K$-means clustering [14, 2]. Two things should be noted: The method to build the inverted index is unsupervised and it is independent from the way subsequent distance approximations are conducted (e.g., while VQ is used to build the index, short lists can be scanned using binary embeddings [14]). In this work, we propose a unifying supervised framework. Both the inverted index and the encoding of features are designed and trained together for improved performance. In the next section, we expose in more detail the existing tools to design IVF/approximate search pipelines, before moving to our proposal in Section 4.

3. Review of image indexing

Image indexing systems are based on two main components: (i) an inverted file and (ii) a feature encoder. In this section we describe how these two main components are used in image indexing systems, thus laying out the motivation for the method we introduce in §4.

Inverted File (IVF)

An inverted file relies on a partition of the database into mutually exclusive bins, a subset of which is searched at query time. The partitioning is implemented by means of VQ [25, 15, 2]: Given a vector $x \in \mathbb{R}^d$ and a codebook $D = [d_k \in \mathbb{R}^d]_{k=1}^N$, the VQ representation of $x$ in $D$ is obtained by solving

$$n = \arg\min_k \|x - d_n\|_2^2,$$

where $n$ is the codeword index for $x$ and $d_n$ its reconstruction. Given a database $\{x_i\}_i$ of image features, and letting $n_i$ represent the codeword index of $x_i$, the database is partitioned into $N$ index bins $B_n$. These bins, stored along with metadata that may include the features $x_i$ or a compact representation thereof, is known as an inverted file. At query time, the bins are ranked by decreasing pertinence $n_1, \ldots, n_N$ relative to the query feature $x^*$ so that

$$\|x^* - d_{n_1}\| \leq \ldots \leq \|x^* - d_{n_N}\|,$$

i.e., by increasing order of reconstruction error. Using this sorting, one can specify a target number of images $T$ to retrieve from the database and search only the first $B$ bins so that $\sum_{k=1}^{B-1} |B_n_k| \leq T \leq \sum_{k=1}^{B} |B_n_k|$.

It is important to note that all existing state-of-the-art indexing methods employ a variant of the above described mechanism that relies on $K$-means-learned codebooks $D$. To the best of our knowledge, ours is the first method to reap the benefits of deep learning to build an inverted file.

Feature encoder

The inverted file outputs a shortlist of images with indices in $\bigcup_{k=1}^{B} B_n_k$, which needs to be efficiently ranked in terms of distance to the query. This is enabled by compact feature encoders that allow rapid distance computations without decompressing features. It is important to note that the storage bitrate of the encoding affects – besides storage cost – search speed, as higher bitrates mean that bins need to be stored in secondary storage, where look-up speeds are a significant burden.

State-of-the-art image indexing systems use feature encoders that employ a residual approach: A residual is computed from each database feature $x$ and its reconstruction $d_n$, obtained as part of the inverted file bin selection in (1):

$$r_n = x - d_n.$$  

This residual is then encoded using a very high resolution quantizer. Several schemes exist [5, 15] that exploit structured quantizers to enable low-complexity, high-resolution quantization, and herein we describe product quantizers and related variants [15, 20, 8]. Such vector quantizers employ a codebook $C \in \mathbb{R}^{d \times K^M}$ with codewords that are themselves additions of codewords from $M$ smaller constituent
The one-hot representation of the encoding look-ups and additions, establishing the functional benefit of formula (6) can be expressed as follows:

$$\sum_{m=1}^{M} c_{m,k} \in \mathbb{R}^{d \times K}, m = 1, \ldots, M,$$

where $C_{m} = [c_{m,k}]_{k} \in \mathbb{R}^{d \times K}, m = 1, \ldots, M$, that are orthogonal (\(\forall m \neq l, C_{m}^{T} C_{l} = 0\)).

Accordingly, an encoding of $r$ in this structured codebook is specified by the indices \((k_1, \ldots, k_M)\) which uniquely define the codeword $c$ from $C$, i.e., the reconstruction of $r$ in $C$. Note that the bitrate of this encoding is $M \log_2(K)$.

**Asymmetric distance computation** Armed with such a representation for all database vectors, one can very efficiently compute an approximate distance between a query $x^*$ and all database features $x \in \{x_i, i \in \cup_{k=1}^{B} B_{n_k}\}$ in top-ranked bins. The residual of $x^*$ for bin $B_n$ is

$$r_n^* = x - d_n,$$

and the approach is asymmetrical in that this uncompressed residual is compared to the compressed, reconstructed residual representation $c$ of the database vectors $x$ in bin $B_n$ using the distance

$$\|r_n^* - c\|_2^2 = \sum_{m=1}^{M} \|r_n^* - c_{m,k}\|_2^2. \quad (6)$$

We define the look-up tables (LUT)

$$z_{n,m} \triangleq \left[ \|r_n^* - c_{m,k}\|_2^2 \right]_k \in \mathbb{R}^K \quad (7)$$

containing the distances between $r_n^*$ and all codewords of $C_m$. Building these LUTs enables us to compute (6) using $\sum_{m=1}^{M} z_{n,m}[k_m]$, an operation that requires only $M$ table look-ups and additions, establishing the functional benefit of the encoding \((k_1, \ldots, k_M)\).

To gain some insight into the above encoding, consider the one-hot representation $b_m$ of the indices $k_m$ given by

$$b_m = [l = k_m]_l \in \mathbb{K}_K, \quad (8)$$

where \([\cdot]_l\) denotes the Iverson brackets and

$$\mathbb{K}_K \triangleq \{a \in \{0, 1\}^K, \|a\|_1 = 1\}. \quad (9)$$

Using stacked column vectors

$$b = \text{COL}(b_1, \ldots, b_M) \in \mathbb{K}_K^M \quad (10)$$

$$z_n = \text{COL}(z_{n,1}, \ldots, z_{n,M}) \in \mathbb{R}_{++}^{M \times K}, \quad (11)$$

distance (6) can be expressed as follows:

$$\|r_n^* - c\|_2^2 = z_n^T b. \quad (12)$$

Namely, computing approximate distances between a query $x^*$ and the database features $x \in \{x_i, i \in B_n\}$ amounts to computing an inner-product between a bin-dependent mapping $z_{n,m} \in \mathbb{R}^{M \times K}$ of the query feature $x^*$ and a block-structured binary code $b \in \mathbb{K}_K^M$ derived from $x$. A search then consists of computing all such approximate distances for the $B$ most pertinent bins and then sorting the corresponding images in increasing order of these distances.

It is worth noting that most of the recent supervised binary encoding methods \([27, 28, 30, 17, 18, 19, 7]\) do not use structured binary codes of the form $b$ in (12). The main exception being StuBiC \([13]\), which further uses a sorting score that is an inner product of the same form as (12).

### 4. A complete indexing pipeline

The previous section established how state-of-the-art large-scale image search systems rely on two main components: an inverted file and a functional residual encoder that produces block-structured binary codes. While compact binary encoders based on deep learning have been explored in the literature, inverted file systems continue to rely on unsupervised $K$-means codebooks.

In this section we first revisit the StuBiC encoder \([13]\), and then show how it can be used to implement a complete image indexing system that employs deep learning methodology both at the IVF stage and compact encoder stage.
4.1. Block-structured codes

The SUBiC encoder in Fig. 1 was the first to leverage supervised deep learning to produce a block-structured code of the form \( b \in \mathcal{K}_K^M \) in (10). At learning time, the method relaxes the block-structured constraint. Letting

\[
\Delta_K = \{ a \in \mathbb{R}_+^K \text{ s.t. } \sum_k a[k] = 1 \}
\]

(13)

denote the convex hull of \( \mathcal{K}_K \), said relaxation

\[
\tilde{b} \in \Delta_K^M
\]

(14)

is enforced by means of a fully-connected layer of output size \( KM \) and ReLU activation with output \( z \) that is fed to a block softmax non-linearity that operates as follows: Let \( z_m \) denote the \( m \)-th block of \( z \in \mathbb{R}^{KM} \) such that \( z = \text{COL}(z_1, \ldots, z_M) \). Likewise, let \( \tilde{b}_m \in \Delta_K \) denote the \( m \)-th block of the relaxed code \( \tilde{b} \in \Delta_K^M \). The block softmax non-linearity operates by applying a standard softmax non-linearity to each block \( z_m \) of \( z \) to produce the corresponding block \( \tilde{b}_m \) of \( \tilde{b} \):

\[
\tilde{b}_m = \left[ \exp(z_m[k]) \sum_l \exp(z_m[l]) \right]_k.
\]

(15)

At test time, the block-softmax non-linearity is replaced by a block one-hot encoder that projects \( \tilde{b} \) unto \( \Delta_K^M \). In practice, this can be accomplished by means of one-hot encoding of the index of the maximum entry of \( z_m \):

\[
b_m = \left[ [k = \text{argmax}(z_m)] \right]_k.
\]

(16)

The approach of [13] introduced two losses based on entropy that enforce the proximity of \( b \) to \( \mathcal{K}_K^M \). The entropy of a vector \( p \in \Delta_K \), defined as

\[
E(p) = \sum_{k=1}^K p[k] \log_2(p[k]),
\]

(17)

has a minimum equal to zero for deterministic distributions \( p \in \mathcal{K}_K \), motivating the use of the entropy loss

\[
\ell_E(b) = \frac{1}{M} \sum_{m=1}^M E(\tilde{b}_m)
\]

(18)

to enforce the proximity of the relaxed blocks \( \tilde{b}_m \) to \( \mathcal{K}_K \). This loss on its own, however, could lead to situations where only some elements of \( \mathcal{K}_K \) are favored (c.f. Fig. 2), meaning that only a subset of the support of \( b_m \) is used.

Yet entropy likewise has a maximum of \( \log_2(K) \) for uniform distributions \( p = \frac{1}{K} \). This property can be used to encourage uniformity in the selection of elements of \( \mathcal{K}_K \) by means of the negative batch entropy loss, computed for a batch \( A = \{ b^{(i)} \} \), of size \( |A| \) using

\[
\ell_B(A) = -\sum_{m=1}^M E\left( \frac{1}{|A|} \sum_i \tilde{b}_m^{(i)} \right).
\]

(19)

For convenience, we define the SUBiC loss computed on a batch \( A \) as the weighted combination of the two entropy losses, parametrized by the hyper-parameters \( \mu, \gamma \in \mathbb{R}_+ \):

\[
\ell_{S,\gamma}^\mu(A) = \frac{\mu}{|A|} \sum_{b \in A} \ell_E(b) + \gamma \ell_B(A).
\]

(20)

It is important to point out that, unlike the residual encoder described in §3, the SUBiC approach operates on the feature vector \( x \) directly. Indeed, the SUBiC method is only a feature encoder, and does not implement an entire indexing framework.

4.2. A novel indexing pipeline

We now introduce our proposed network architecture that uses the method of [13] described above to build an entire image indexing system. The system we design implements the main ideas of the state-of-the-art pipeline described in §3.
Our proposed network architecture is illustrated in Fig. 3. The input to the network is the feature vector \( z \) consisting of activation coefficients obtained by running a given image \( I \) through a CNN feature extractor. We refer to this feature extractor as the base CNN of our system.

Similarly to the design philosophy described in §3, our indexing system employs an IVF and a residual feature encoder. Accordingly, the architecture in Fig. 3 consists of two main blocks, Bin selection and Encoder, along with a Residual block that links these two main components.

**Bin selection** The first block, labeled Bin selection in Fig. 1, can be seen as a StBtC encoder employing a single block \( (i.e. M = 1) \) of size \( N \), with the block one-hot encoder substituted by an argmax operation. The block consists of a single fully-connected layer with weight matrix \( W_1 \) and ReLU activation followed by a second activation using softmax. When indexing a database image \( I \), this block is responsible for choosing the bin \( B_n \) that \( I \) is assigned to, using the argmax of the coefficients \( z^* \).

Given a query image \( I^* \), the same binning block is responsible for sorting the bins in decreasing order of pertinence \( B_n, \cdots B_{n_M} \) using the coefficients \( z^{*\prime} \in \mathbb{R}^N_+ \) so that
\[
z^{*\prime}[n_1] \geq \cdots \geq z^{*\prime}[n_N],
\]
in a manner analogous to (2).

**Residual feature encoding** Inspired by the residual encoding approach described in §3, we consider a block analogous to the residual computation of (3) and (5). The approach consists of building a vector (denoting ReLU as \( \sigma \))
\[
R_2 \sigma(R_1 \tilde{b}'),
\]
that is analogous to the reconstruction \( d_n \) of \( x \) obtained from the encoding \( n \) following the IVF stage (cf. (1) and discussion thereof), and subtracts it from a linear mapping of \( x \):
\[
r = Qx - R_2 \sigma(R_1 \tilde{b}').
\]

Besides the analogy to indexing pipelines, one other motivation for the above approach is to provide information to the subsequent feature encoding from the IVF bin selection stage \( (i.e. \tilde{b}') \) as well as the original feature \( x \). For completeness, as illustrated in Fig. 3, we also consider architectures that override this residual encoding block, setting \( r = x \) directly.

The final stage consists of an \( M \)-block StBtC encoder operating on \( r \) and producing test-time encodings \( b \in \mathbb{K}_N^M \), and training-time relaxed encoding \( \tilde{b} \in \mathbb{K}_N^M \). Note that, unlike the residual approach described in §3, our approach does not incur the extra overhead required to compute LUTs using (7).

**Searching** Given a query image \( I^* \), it is first fed to the pipeline in Fig. 3 to obtain (i) the activation coefficients \( z'^* \) at the output of the \( W_1 \) layer and (ii) the activation coefficients \( z^* \) at the output of the \( W_2 \) layer. The IVF bins are then ranked as per (21) and all database images \( \{I, i \in \bigcup_{k=1}^B B_{n_k}\} \) in the \( B \) most pertinent bins are sorted, based on their encoding \( b_i \), according to their score
\[
z^{*\prime}Tb_i.
\]

**Training** We assume we are given a training set \( \{(I^{(i)}, y^{(i)})\}_i \) organized into \( C \) classes, where label \( y^{(i)} \in \{1, \ldots, C\} \) specifies the class of the \( i \)-th image. Various works on learning for retrieval have explored the benefit of using ranking losses like the triplet loss and the pair-wise loss as opposed to the cross-entropy loss successfully used in classification tasks [11, 27, 28, 30, 17, 18, 19, 7, 5]. Empirically, we have found that the cross-entropy loss yields good results in the retrieval task, and we adopt it in this work.

Given an image belonging to class \( c \) and a vector \( p \in \Delta_C \) that is an estimate of class membership probabilities, the cross-entropy loss is given by (the scaling is for convenience of hyper-parameter cross-validation)
\[
\ell(p, c) = -\frac{1}{\log_2 C} \log_2 p[c].
\]

Accordingly, we train our network by enforcing that the relaxed block-structured codes \( \tilde{b}' \) and \( \tilde{b} \) are good feature vectors that can be used to predict class membership. We do so by feeding each vector to a soft-max classification layer (layers \( C_1 \) and \( C_2 \) in Fig. 3, respectively), thus producing estimates of class membership \( s' \) and \( s \) in \( \Delta_C \) (cf. Fig. 3) from which we derive two possible task-related losses. Letting \( T \) denote a batch specified as a set of training-pair indices, these two losses are
\[
L_{1,\alpha} = \frac{1}{|T|} \sum_{i \in T} \left[ \alpha \ell(s'^{(i)}, y^{(i)}) + \ell(s^{(i)}, y^{(i)}) \right]
\]
and
\[
L_2 = \frac{1}{|T|} \sum_{i \in T} \ell(s'^{(i)}, s^{(i)}, y^{(i)}),
\]
where the scalar \( \alpha \in \{0, 1\} \) is a selector variable. In order to enforce the proximity of the \( \tilde{b}' \) and \( \tilde{b} \) to \( \mathbb{K}_N \) and \( \mathbb{K}_M^M \), respectively, we further employ the loss
\[
\Omega_H = \ell_0^{\mu_1, \gamma_1}(\{\tilde{b}'^{(i)}\}_{i \in T}) + \ell_0^{\mu_2, \gamma_2}(\{\tilde{b}^{(i)}\}_{i \in T}),
\]
which depends on the four hyper-parameters \( H = \{\mu_1, \gamma_1, \mu_2, \gamma_2\} \) (we discuss heuristics for their selection if §3).

Accordingly, the general learning objective for our system is
\[
F_* = L_* + \Omega_H,
\]
and we consider three variants thereof:
We compare all three variants of our proposed indexing system against two existing base-
lines, as well as a straightforward attempt to use deep hashing as an IVF system:

(IIVF-PQ) This approach uses an inverted file with $N = 4096$ bins followed by a residual PQ encoder with $M = 8$ blocks and constituent codebooks of size $K = 256$ (c.f. (4)), resulting in an 8-byte feature size. The search employs asymmetric distance computation. During retrieval, the top $B = 2^n$ lists are retrieved, and, for each $n = 1, 2, \ldots$ the average mAP and average aggregate bin size $T$ are plotted.

(IMI-PQ) The Inverted Multi-Index (IMI) [2] extends the standard IVF by substituting a product quantizer with $M = 2$ and $K = 4096$ in place of the vector quantizer. The resulting IVF has more than 16 million bins, meaning that, for practical testing sets (containing close to 1 million images), most of the bins are empty. Hence, when employing IMI, we select short-list sizes $T$ for which to compute average mAP to create our plots. Note that, given the small size of the IMI bins, the computation of the look-up tables $z_n$ (c.f. (7)) represents a higher cost per-image for IMI than for IVF. Furthermore, the fragmented memory reads required can have a large impact on speed relative to the contiguous reads implicit in the larger IVF bins.

(DSH-SUBIC) In order to explore possible approaches to include supervision in the IVF stage of an indexing system, we further considered using the DSH deep hash code [19] as a bin selector, carefully selecting the regularization parameter to be 0.03 by means of cross-validation. We train this network to produce 12-bit image representations corresponding to $N = 4096$ IVF bins, where each bin has an associated hash code. Images are indexed using their DSH hash, and at query
Table 1. Instance retrieval with encoded features. Performance (mAP) comparison using 64-bit codes, first row shows reference results with original uncompressed features. When bounding box information is used for Oxford5K dataset, the performance degrades for both PQ and SUbIC, shown in column Oxford5K*, as both are trained on full images.

Large-scale indexing Fig 4 shows the mAP performance versus average number of retrieved images \( T \) for all three variants as well as the baselines described above. Note that the number of retrieved images is a measure of complexity, as for IVF, the time complexity of the system is dominated by the approximate distance computations in (12). For IMI, on the other hand, there is a non-negligible overhead on top of the approximate distance computation related to the large number of bins, as discussed above.

We present results for three datasets (Oxford5K, Paris6K, Holidays), on three different database scales (the original dataset, and when also including noise datasets of \( 100K \) and \( 1M \) images). Note that on Oxford5K and Paris6K, both SUbIC-I and SUbIC-J enjoy large advantages relative to all three baselines – at \( T = 300 \), the relative advantage of SUbIC-I over the IMI-PQ is 19% at least. SUbIC-R likewise enjoys an advantage on the Paris6K dataset, and performs comparably to the baselines on Oxford5K.

On Holidays SUbIC-I outperforms IVF-PQ by a large margin (18% relative), but not outperform IMI-PQ. As dis-
Concerning the poor performance of $\text{SUIC-R}$ on Holidays, we believe this is due to poor generalization ability of the system because of the three extra fully-connected layers.

**IMI extension** Given the high performance of IMI for the Holidays experiments in Fig. 4, we further consider an IMI variant of our $\text{SUIC-I}$ architecture. To implement this approach, we learn a $\text{SUIC}-(2,4096)$ encoder (with $\mu = 4$ and $\gamma = 5$). Letting $z'_{m}$ denote the $m$-th block of $z'$, the $(k, l) \in (1, \ldots, 4096)^2$ bins of $\text{SUIC-IMI}$ are sorted based on the score $z'_1[k] + z'_2[l]$. For fairness of comparison, we use the same $\text{SUIC}-(8, 256)$ feature encoder for all methods including the baselines, which are IVF and IMI with unsupervised codebooks (all methods are non-residual). The results, plotted in Fig. 5, establish that, for the same number of bins, our method can readily outperform the baseline IMI (and IVF) methods. Furthermore, given that we use the best performing feature encoder ($\text{SUIC}$) for all methods, this experiment also establishes that the $\text{SUIC}$ based binning system that we propose outperforms the unsupervised IVF and IMI baselines.

**Category retrieval** For completeness, we also carry out experiments in the category retrieval task which has been the main focus of recent deep hashing methods [27, 28, 30, 17, 18, 19, 7]. In this task, a given query image is used to rank all database images, with a correct match occurring for database images of the same class as the query image. For category retrieval experiments, we use VGG-M-128 base features [24], which have established good performance for classification tasks, and a $\text{SUIC}-(1, 8192)$ for bin selection. We use the ImageNet training set (1M+ images) to train, and the test (training) subsets of Pascal VOC and Caltech-101 as a query (respectively, database) set. We
present results for this task in Fig. 6. Note that, unlike the Holidays experiments in Fig. 4, SuperIC-R performs best on Caltech-101 and equally well to SuperIC on Pascal VOC, a consequence of the greater size and diversity of the ImageNet datasets relative to the Landmarks dataset.

6. Conclusion

We present a full image indexing pipeline that exploits supervised deep learning methods to build an inverted file as well as a compact feature encoder. Previous methods have either employed unsupervised inverted file mechanisms, or employed supervision only to derive feature encoders. We establish experimentally that our method achieves state of the art results in large scale image retrieval.

References

[1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In Proc. Int. Conf. Computer Vision, 2016.

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

[3] A. Babenko and V. Lempitsky. Additive quantization for extreme vector compression. In Proc. Conf. Comp. Vision Patte...r Rec., 2014.

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

[5] C. Bilen, J. Zepeda, and P. Perez. The CNN News Footage Datasets : Enabling Supervision in Image Retrieval. EU-SIPCO, 2016.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

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

[13] H. Jain, J. Zepeda, P. Pérez, and R. Gribonval. SuperIC: A supervised, structured binary code for image search. In Proc. Int. Conf. Computer Vision, 2017.

[14] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. Proc. Europ. Conf. Computer Vision, 2008.

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

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

[17] 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.

[18] 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.

[19] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In Proc. Conf. Comp. Vision Pattern Rec., 2016.

[20] M. Norouzi and D. Fleet. Cartesian k-means. In Proc. Conf. Comp. Vision Pattern Rec., 2013.

[21] 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.

[22] 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.

[23] 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.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014.

[25] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In Proc. Int. Conf. Computer Vision, 2003.

[26] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. arXiv preprint arXiv:1408.2927, 2014.
[27] 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. 2, 3, 5, 8

[28] 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. 2, 3, 5, 8

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

[30] 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. 2, 3, 5, 8