Unsupervised Hashing with Neural Trees for Image Retrieval and Person Re-Identification

Niki Martinel
University of Udine
Udine, Italy
niki.martinel@uniud.it

Gian Luca Foresti
University of Udine
Udine, Italy
gianluca.foresti@uniud.it

Christian Micheloni
University of Udine
Udine, Italy
christian.micheloni@uniud.it

ABSTRACT

Recent vision studies have shown that learning compact codes is of paramount importance to improve the massive data processing while significantly reducing storage footprints. This has recently yielded to a surge of effort in learning compact and robust hash functions for image retrieval tasks. The majority of the existing literature has been devoted to the exploration of deep hash functions, typically under supervised scenarios. Unsupervised hashing methods have been less attractive due to the their difficulty in achieving satisfactory performance for the same objective. In this work, we propose a simple yet effective unsupervised hashing framework, which exploits the powerful visual representation capabilities of deep architectures and combines this within a tree structure involving a multi-path scheme. The key advantage of the proposed method is the ability to bring in a *divide-and-conquer* approach to reduce the complexity of the classification problem at each node of the tree without the need of labeled data. To validate the proposed solution, experimental results on two benchmark datasets for image retrieval and person re-identification have been computed.

KEYWORDS

Hash Encoding, Neural Tree, Hierarchical Learning

ACM Reference Format:
Niki Martinel, Gian Luca Foresti, and Christian Micheloni. 2018. Unsupervised Hashing with Neural Trees for Image Retrieval and Person Re-Identification: . In *International Conference on Distributed Smart Cameras (ICDSC '18), September 3–4, 2018, Eindhoven, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3243394.3243687

1 INTRODUCTION

Being able to fetch a list of relevant samples starting from a visual similarity search is a problem of paramount importance for information retrieval applications [17]. In particular, when such a process has to be conducted on large-scale visual sets, hashing techniques are exploited. This attractive approach generally digest high-dimensional visual features to generate a binary vector lying in a similarity-preserved low-dimensional Hamming space. The encoded hashes significantly reduce both the storage requirements and the computational burden.

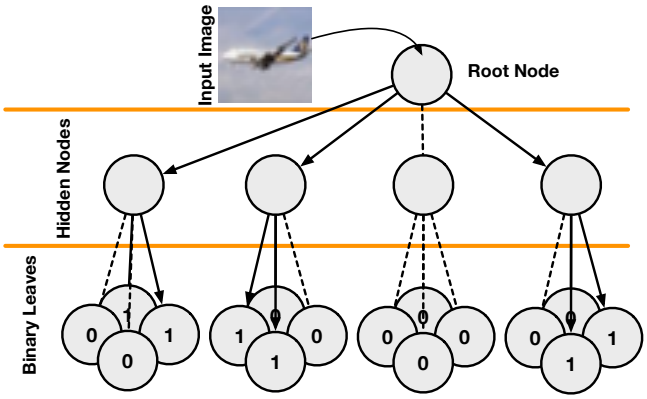

Figure 1: Illustration of the proposed tree-based architecture used as hash generator. An image is the input to the whole architecture, which grows by adding hidden child nodes as long as the stop criteria is not met. Since the input datum can be routed to multiple child nodes, we use the notification of leaf nodes receiving the input as a binary answer which is finally concatenated to build the hash code.

As discussed in recent surveys [29, 30], a surge of effort has been recently spent to propose novel hashing methods. The existing literature can be categorized into two main groups: data-independent and data-dependent methods.

Approaches belonging to the former category rely on random projections to create hash functions generating compact binary codes. The Iterative Quantization (ITQ) algorithm [3] learns an orthogonal rotation matrix by minimizing the quantization loss while mapping the data generated by PCA (Principal Component Analysis) projections to binary codes. ISpherical Hashing (SpH) [5] exploits hypersphere-based hashing function maps spatially coherent data points to similar binary codes. In [16], the table construction is treated as a selection problem over a set of candidate hash functions while in Spectral Hashing (SH) [31] the hash function learning problem is formulated as a particular form of graph partition.Minimal Loss Hashing (MLH) [23] adopts a pairwise hinge-like loss function and minimizes its upper bound to learn binary codes.

This research was partially supported by the Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 765866.

ICDSC '18, September 3–4, 2018, Eindhoven, Netherlands

ACM ISBN 978-1-4503-6511-6/18/09…$15.00
https://doi.org/10.1145/3243394.3243687

Binary Reconstructive Embedding (BRE) [9] utilizes pairwise relations between samples and minimizes the squared error between the original normalized Euclidean and Hamming distances.

Methods within the latter group leverage on the available training data to learn hash functions either in an unsupervised (e.g., [27, 32]) or supervised fashion (e.g., [2, 14]). Traditional supervised hashing methods aim to preserve the semantic similarity in Hamming space. In [15], the compact binary code is learned by minimizing the Hamming distances on similar pairs and simultaneously maximizing the Hamming distances on dissimilar pairs. In [11], a flexible yet simple framework is proposed to accommodate different types of loss functions and hash functions.

Supervised hashing methods include deep-learning inspired techniques as well. In [33], the hash learning function problem is posed as a regularized similarity learning task. In [13], a novel neural network is developed to learn binary codes preserving the non-linear relationship of samples.

For unsupervised hashing, the methods based on deep neural networks [1, 24] only involve the point-wise constraints, due to the lack of labels or class information. As a pioneering work using deep learning techniques for unsupervised hashing, semantic hashing [24] applies the stacked Restricted Boltzmann Machine (RBM) to learn compact binary codes for visual search.

A crucial issue with these deep learning strategies for hash function learning is that they have to deal with binary codes. To handle the NP-hard mixed-integer optimization problem, a large majority of the literature relaxes such a constraint during the learning process. Thus, the continuous codes are learned first, then binarization is applied (e.g., with thresholding) to obtain the final hash. However, the solution can be suboptimal, i.e., the binary codes resulting from thresholded continuous codes could be inferior to those that are obtained by including the binary constraint in the learning.

In addition to this tasks-specific issues, CNN-based solutions have a significant drawback regarding the definition of the architecture. Indeed, there is currently no fixed *a priori* rule to decide, for any given problem, the architecture of a deep net (i.e., number of hidden layers and nodes in each of these).

To overcome some of these limitations, we introduce a novel hybrid architecture (see Figure 1) originating from Neural Trees (NTs). The main motivation for the development of this new architecture came from the search for a training algorithm able to learn the structure of the architecture rather than requiring its upfront definition (e.g., [20, 22]). Furthermore, to keep the powerful features of the hierarchical visual learning frameworks, within each node of the tree-based solution we leverage on the power of deep neural networks as feature extractors and exploit such neural representations in an unsupervised learning technique used for routing patterns to child nodes. The hash code is obtained by combining the binary notifications generated by each leaf node receiving the input datum.

Contributions.

(i) We propose to use an hybrid neural architecture as a hash function for unsupervised hashing. The tree-based approach allows us to severely limit the suboptimal and inefficient trial and error process in deciding the deep net architecture while still keeping the desirable non-linear mapping. To the best of our knowledge, our approach is the first unsupervised hashing method that uses a neural tree as hash function.

(ii) In order to efficiently learn the neural tree architecture for unsupervised hashing, within each node of the tree, we apply a two-step strategy. We first exploit the power of deep learning architectures to obtain the neural feature representation of the input datum. Then, we leverage on a clustering solution to route patterns that are close to each other in the neural feature space to the same child node.

(iii) We introduce a multiple routing approach based on the confidence of each cluster on the input features. By doing this, we aim to both overcome the issue generated by routing a pattern to a single, maybe wrong, child as well as to increase the number of training patters that can be exploited to learn each in-node model.

(iv) For binary code inference, we exploit the multiple-routing solution and construct the binary code upon notification of leaves receiving the input pattern.

To substantiate our contributions we have conducted the experiments on two publicly available benchmark datasets: one for image retrieval and the other for person re-identification. Results demonstrate that the proposed approach obtains comparable performance with state-of-the-art methods that involve more complicated and computationally demanding training procedure.

2 UNSUPERVISED HASH LEARNING

2.1 Definitions

Consider a hash coding problem with input and (finite) output spaces given by $\mathcal{I}$ and $\mathcal{Y}$, respectively. A neural tree is a tree-structured architecture consisting of *internal nodes* and *terminal nodes*. Let $\psi \in \mathbf{\Psi}$ denote an internal node of the tree and let $\lambda \in \mathbf{\Lambda}$ represent a terminal nodes of the tree. The aim is to learn the tree structure and its internal route node models such that, for any given image $\mathbf{I} \in \mathcal{I}$, the tree generates a hash code $\mathbf{h}$ of length m. The pipeline of the proposed solution is shown in 2.

To obtain the hash code, each routing node $\psi \in \mathbf{\Psi}$ sees a set of training patterns $\mathcal{X}^{(\psi)} = \{\mathbf{x}_j^{(\psi)}\}_{j=1}^{|\mathcal{X}^{(\psi)}|}$ with each $\mathbf{x}_j^{(\psi)} \in \mathbb{R}^{M \times N \times d}$. These are first exploited to obtain their neural feature representation $f(\mathbf{x}^{(\psi)}; \mathcal{W}^{(\psi)}) : \mathcal{X}^{(\psi)} \rightarrow \mathcal{F}^{(\psi)}$ controlled by a set of pre-trained parameters $\mathcal{W}^{(\psi)}$. The so obtained visual features are later considered to learn the parameters $\mathbf{\Theta}^{(\psi)}$ controlling the routing function $\pi^{(\psi)}(\cdot; \mathbf{\Theta}^{(\psi)}) : \mathcal{F}^{(\psi)} \rightarrow [1, k]$. When a sample $\mathbf{x}^{(\psi)} \in \mathcal{X}^{(\psi)}$ reaches a routing node ψ it will be sent to at least one of its child nodes on the basis of $\pi^{(\psi)}(f(\mathbf{x}^{(\psi)}; \mathcal{W}^{(\psi)}); \mathbf{\Theta}^{(\psi)})$.

Each leaf node $\lambda \in \mathbf{\Lambda}$ generates a binary label $h_\lambda \in \{0, 1\}$. All together, the binary leaves yield to the hash code $\mathbf{h} = \{h_\lambda\}_{\lambda \in \mathbf{\Lambda}}$.

2.2 Internal Nodes

Each internal node of the tree $\psi \in \mathbf{\Psi}$ is responsible of a two-step process. It first computes the neural feature representation of the input data. Then, it exploits the obtained representations to learn the parameters $\mathbf{\Theta}^{(\psi)}$ controlling the routing function. In the following, the two steps are described in details. To ease the notation, the (ψ) superscript is omitted.

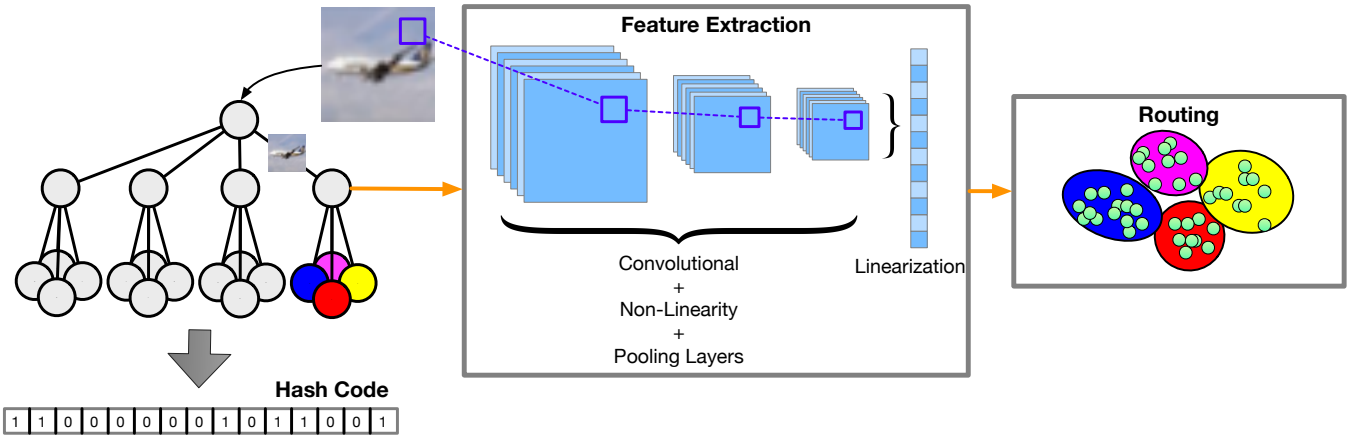

Figure 2: Illustration of the proposed hash encoding pipeline. On the left hand side, an overview of the introduced tree structure is depicted. On the right hand side, an exploded view of an internal node is depicted. This includes a CNN-based feature extraction mechanism as well as a color-coded clustering approach that is exploited to route input patterns towards child nodes. The tree leaves are responsible of creating the final hash code.

2.2.1 Feature Extraction. Inspired by the wide success of deep neural networks as generic visual feature detectors, we consider f being a deep neural network composed by a stack of convolutional feature detectors, non-linearity mappings and a pooling operators. The weights of the stacked convolutional feature detectors are parameterized by $\mathcal{W}^{(\psi)}$. Such parameters are obtained by exploitation of state-of-the-art deep neural networks pre-trained on image recognition tasks with supervised data (e.g., VGG [26]).

Specifically, to get the neural features we adopt a CNN feature extractor which uses a set of n kernel weights $\mathbf{W} = \{\mathbf{w}_i\}_{i=1}^{n} \in \mathcal{W}^{(\psi)}$ to produce, for any given input pattern $\mathbf{x}$, the i-th feature map obtained as

$$\hat{\mathbf{x}}(a,b,i) = \sum_{u=1}^{w}\sum_{v=1}^{w}\sum_{t=1}^{d}\mathbf{x}(a+u,b+v,t)\mathbf{w}_i(u,v,t) \tag{1}$$

for $a \in \{0,\cdots,M-w-1\}$ and $b \in \{0,\cdots,N-w-1\}$, where w denotes the width/height of the receptive field. The resulting feature maps are then input to the non-linearity function and subsequently to the pooling operator as defined by the considered CNN architecture. The output of a stack of such layers is the neural representation for the input datum $\mathbf{x}$. All these steps combined can be summarized as $\hat{\mathbf{x}} = f(\mathbf{x};\mathcal{W})$.

2.2.2 Routing Patterns. In standard decision trees, each node evaluates a (generally parameterized) split function that is in charge of routing input patterns towards one of two children. We remove such a constraint and let an input pattern be forwarded to *at least* one its k child nodes. Such a relaxation allows us to increase the number of patters that reach each children, hence to enlarge the training set that can be exploited within each node. In addition, having a same pattern flowing towards more than a single leaf node will open to multiple "decision answers" –over the same pattern– which can be considered to build the final hash code.

We propose to use a routing function that depends on the relation of the considered neural representations. Following such an idea, we let the $\pi(\cdot;\Theta)$ be a two step function that (i) clusters the input patterns through k-means, then (ii) determines how patterns are routed to one or more child nodes. K-means has been chosen over more complex non-linear alternatives (e.g., Gaussian Mixtures, DBSCAN) since we aim to capture the data non-linearities through the hierarchical structure, not through the routing procedure *per se*. More formally, given the set of neural features $\hat{\mathcal{X}}^{(\psi)} = \{\hat{\mathbf{x}}_j\}_{j=1}^{|\hat{\mathcal{X}}^{(\psi)}|}$ reaching a specific node ψ, this is used to find the k-means clustering centroids $\{\boldsymbol{\mu}_c\}_{c=1}^{k}$ by solving

$$\arg\min_{\boldsymbol{\mu}} \sum_{c=1}^{k}\sum_{\hat{\mathbf{x}}\in C_c} \|\hat{\mathbf{x}}-\boldsymbol{\mu}_c\|_2^2 \tag{2}$$

where C_c denotes the c-th cluster set that needs to be identified and $\|\cdot\|_2^2$ is the squared ℓ_2-norm applied to elementiwse differences.

The learned cluster centroids can be used to route the input patterns towards the k child nodes. A standard approach would have been to send a given pattern to a *single child* node, indexed by the closest cluster centroid. However, in doing so, we would not consider the contribution of other clusters to the routing decision. In particular, if a pattern lies on the boundary separating two (or more) clusters, it means that the path to be followed is highly uncertain. In light of such an intuition, we propose to capture and exploit such uncertainty through a probabilistic approach.

We begin by converting the dissimilarities of the given pattern with the obtained centroids to probabilities. This is achieved through the exponential decay function resulting in

$$p(\hat{\mathbf{x}}|\boldsymbol{\mu}_c) = \frac{\exp\left(-\|\hat{\mathbf{x}}-\boldsymbol{\mu}_c\|_2^2\right)}{\sum_{i=1}^{k}\exp\left(-\|\hat{\mathbf{x}}-\boldsymbol{\mu}_i\|_2^2\right)} \tag{3}$$

with $\sum_{c=1}^{k} p(\mathbf{x}|\boldsymbol{\mu}_c) = 1$. Then, we hypothesize that the obtained probabilities are Normally distributed, hence exploit the distribution characteristics to send a pattern to the c-th child iff

$$p(\hat{\mathbf{x}}|\boldsymbol{\mu}_c) > \max_{c}(p(\hat{\mathbf{x}}|\boldsymbol{\mu}_c)) - \lambda\sigma(p(\hat{\mathbf{x}})) \tag{4}$$

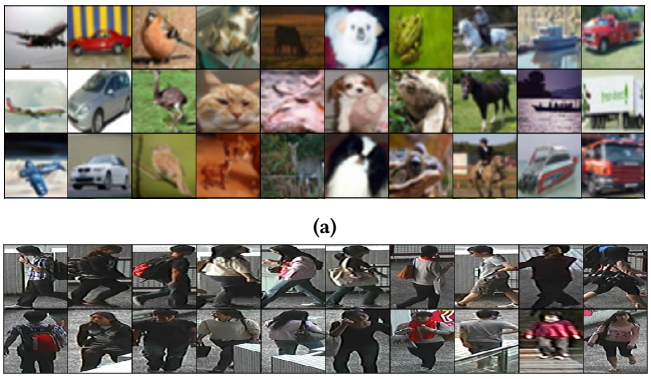

Figure 3: Sample images of ten different objects appearing in (a) CIFAR-10, and (b) CUHK03 datasets. Each row highlights the intra-class variations for the same category (one per column). *(Best viewed in color)*

where $\lambda = 2$ (we considered the standard 95% rule of the Normal distribution), and $\max_c(p(\hat{\mathbf{x}}|\boldsymbol{\mu}_c))$ and $\sigma(p(\hat{\mathbf{x}}))$ denote the maximum probability and the standard devision of the distribution over p, respectively.

2.3 Terminal Nodes

With the feature representation function and the clustering procedure we have defined an approach to route a pattern $\mathbf{x}^{(\psi)}$ reaching a particular internal node ψ to its children. The tree structure grows by adding internal nodes as long as a stop criterion is not satisfied. In this work, such a criterion is the maximum depth of the tree –denoted as γ. This, together with the number of child nodes k, allows us to control the number of terminal nodes, hence the hash encoding dimension m.

More precisely, when an internal node satisfies the stop criterion, this is converted into a terminal node λ. This will result in the tree having $|\boldsymbol{\Lambda}| = m = k^{\gamma}$ leaves.

For any given sample $\mathbf{x}$, its hash code $\mathbf{h}$ is obtained by traversing the tree in a top-down fashion starting from the root node. All the node-dependent learned parameters (i.e., $\mathcal{W}^{(\psi)}$ and $\boldsymbol{\Theta}^{(\psi)}$) are exploited to route the sample towards the leaves. Thanks to the proposed routing procedure the sample can reach more than a single leaf node. We use this particular feature to obtain a binary answer from each leaf node. That is, any leaf node λ will generate a positive binary output (i.e.,$h = 1$) if it receives the sample. Otherwise, a negative binary output (i.e., $h = 0$) is generated. The hash code is constructed by collecting the binary outputs from all the leaves as $\mathbf{h} = \{h_i\}_{i=1}^{m}$.

3 EXPERIMENTAL RESULTS

3.1 Datasets

To validate the proposed approach, experimental evaluations on two benchmark datasets have been carried out.

CIFAR-10[1]. The CIFAR-10 [8] is a challenging dataset containing color images belonging to 10 different object classes (see Figure3(a)). Each class comes with 6,000 32×32 samples, leading to a total of 60,000 images. The dataset is already partitioned into a training set with 50,000 samples and a test set having 10,000 images. To provide a fair comparison with existing solutions, we followed a common approach [32] and for each class in the test set randomly selected 100 samples as the query set (1,000 images in total). The remaining portion of the test is used to extend the training set, thus forming a dataset of 59,000 images.

CUHK03[2]. The CUHK03 [10] dataset is one of the largest and most challenging Re-Identification datasets (see Figure 3(b)). It contains 13,164 images of 1,360 pedestrians acquired by six disjoint cameras. Each person has been observed by two disjoint camera views and has an average of about 5 images in each view. To run the experiments, we followed the same procedure as in [10] and used the 20 provided trials with the manually labeled detections. Each of these splits the data into a training set and a test set containing 1,160 and 100 persons, respectively.

3.2 Evaluation Protocol

Evaluation of image retrieval approaches (e.g., [2, 30, 32, 35]) is generally performed by showing the mean Average Precision (*mAP*) as well as the precision@N retrieved samples. The mAP is a performance indicator that is independent from the dataset size and is obtained as

$$\text{mAP} = \frac{1}{Q}\sum_{i=1}^{|Q|}\left(\frac{1}{|\mathcal{R}_i|}\sum_{j=1}^{\mathcal{G}_i} P_i(j)\mathbb{1}_{\{\mathcal{G}_i(j)\in\mathcal{R}_i\}}\right) \tag{5}$$

where, Q is the query image set, $\mathcal{R}_i$ is the set of relevant images and $\mathcal{G}_i$ is the ranked set of retrieved images. $\mathcal{G}_i(j)$ and $P_i(j)$ represents the j-th retrieved image and the precision at j for the i-th query, respectively. $\mathbb{1}_{\{\cdot\}} \in \{0, 1\}$ is the indicator function. The precision@N measure gives the percentage of ground truth images among *top–N* retrieved samples. We followed such a common protocol and report on the results of our method using such indicators also considering different hash code lengths (refer to the next section for details).

3.3 Experimental and Implementation Settings

The majority of the existing image retrieval approaches tweak their hyperparameters to the specific dataset (e.g., [32]). In our evaluation, we have decided not to do so to provide a more generic framework.

Feature Extraction. We exploited an available deep neural network for feature extraction, namely the VGG16 [26] architecture[3]. At each specific depth of the tree, for any given node, we select a subset of the the aforementioned architectures layers dependently on the maximum depth of the considered tree (i.e.,γ). Details on the selected layers are given in Table 1.

Routing. To identify cluster centroids, k-means is run for at maximum 100 iterations. To allow a fair comparison with the results

[1] https://www.cs.toronto.edu/~kriz/cifar.html
[2] http://www.ee.cuhk.edu.hk/~xgwang/CUHK_identification.html
[3] Among the plethora of available architectures, we have selected the VGG16 to have a more fair comparison with existing methods using a similar network. We remind to future works the exploitation of more powerful yet less computationally demanding architectures.

Table 1: Selected layers of the VGG-16 architecture exploited for feature extraction.

Depth	Hash Code Length (m)	Internal Nodes VGG-16 Layers
1	4	64x[Conv3x3] → 64x[Conv3x3] → [MaxPool2x2] → 128x[Conv3x3] → 128x[Conv3x3] → [MaxPool2x2]
2	16	256x[Conv3x3] → 256x[Conv3x3] → 256x[Conv3x3] → [MaxPool2x2]
3	64	512x[Conv3x3] → 512x[Conv3x3] → 512x[Conv3x3] → [MaxPool2x2]

provided by existing methods (i.e., to provide results with $m \in \{16, 32, 64\}$), we set the number of clusters $k = 4$. Please notice that, since our approach is not designed to directly output a hash code length of $m = 32$ (i.e., directly available hash code lengths are only powers of k), we obtain a hash code of $m = 32$ by first generating a hash code of $m = 64$, then removed those leaves that has the lowest number of training patterns.

3.4 Image Retrieval

We systematically compare our method with 6 state-of-the-art non-deep-learning based methods, namely LSH [7], SH [24], PCAH [28], SPhH [5], KMH [4], PCA-ITQ [3] and 5 existing unsupervised deep-learning-based-solutions. These are DH [13], DA [6], DeepBit [12], UH-BDNN [2] and SADH-L [25]. Notice that the non-deep methods and UH- BDNN use the VGG-19 feature extracted from the last fully connected layer.

In Table 2, we report on the comparison with existing methods on the CIFAR-10 dataset. Results show that with an mAP of 17.32% for $m = 64$, the proposed solution achieves similar results to existing methods. Specifically, we obtain the third-best result. However, it should be noted that methods achieving best performance, namely DeepBit [12] and SADH-L [25], require two/three-stages training procedures that require more computational efforts. We hypothesize that their advantage is due to the fact that, by such procedures, they modify the feature extraction functions (i.e., the involved deep network weights) to adapt to the specific problem

3.5 Person Re-Identification

We evaluate our method using CUHK03 dataset with images resized to 96×96. To compute the performance, the widely used Cumulative Matching Characteristic (CMC) curve [18, 19, 21] has been used.

We report on the comparison with five state-of-the-art hash learning methods BRE [9], MLH [23], KSH [15], DSRH [34], and DRSCH [33] and a deep-learning-based re-identification solution (FPNN [10]). When using traditional hashing learning methods, the 4096D CNN features are extracted from an AlexNet network pre-trained on ImageNet. For DSRH and DRSCH, the parameters of the corresponding deep networks are learned from raw images without any pretraining. Please notice that DSRH, DRSCH, and FPNN are supervised methods which have been trained with training samples coming from the same dataset.

Results in Table 3 show that the proposed solution performs better than existing unsupervised state-of-the-art approaches, even

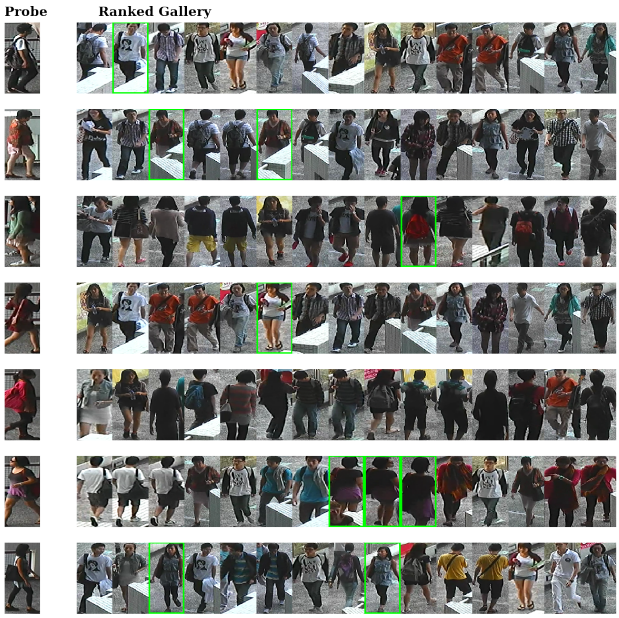

Figure 4: Qualitative ranking performance obtained by the proposed solution on the CUHK03 dataset. Each row show the top 15 retrieved galleries for a given probe. The true match is highlighted in green. *(Best viewed in color)*

with a significantly lower hash code length. To verify the qualitative performance, results in Figure4 have been computed. These show that for each considered query, the retrieved list contains persons that are very similar to each other and to the searched sample. Such retrieval performance might indicate that our solution is able to gather then route similar semantic patterns to a same child node, thus generating similar hash codes.

4 CONCLUSIONS

In this paper we have introduced a novel approach to learn a hash generation function. To achieve such a goal, a hierarchical architecture which grounds on a tree structure has been proposed. The internal nodes of the tree exploits a pre-trained deep neural network to obtain a hierarchical representation of the input datum. This is then used to forward a pattern to multiple child nodes by means of a probabilistic routing function. The hash code is obtained by traversing the tree in a top-down fashion then by collecting the leaf node binary outputs. Experimental results on two benchmark datasets show that our solution achieves state-of-the-art performance.

REFERENCES

[1] Miguel A. Carreira-Perpinan and Ramin Raziperchikolaei. 2015. Hashing with Binary Autoencoders. In *CVPR*.
[2] Thanh Do, Anh Dzung Doan, and Ngai Man Cheung. 2016. Learning to hash with binary deep neural network. *ECCV* 9909 LNCS (2016), 219–234.
[3] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE TPAMI* 35, 12 (2013), 2916–2929.
[4] Kaiming He, Fang Wen, and Jian Sun. 2013. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*. 2938–2945.

Table 2: Retrieval comparisons on the CIFAR-10 dataset with $K \in \{16, 32, 64\}$ bits. Hamming ranking accuracy by mAP and precision@N ($N = 1000$). Best result for each number of bits is in boldface font.

Method	mAP [%]			Precision@N [%]			Publication
	16	32	64	16	32	64	
LSH	12.55	13.76	15.07	16.21	19.10	22.25	ACMComm2008 [7]
SH	12.95	14.09	13.89	14.79	17.87	18.27	JAR2009 [24]
PCAH	12.91	12.60	12.10	18.89	19.35	18.73	CVPR2010 [28]
SphH	13.98	14.58	15.38	20.13	22.33	25.19	CVPR2012 [5]
KMH	13.59	13.93	14.46	20.28	21.97	22.80	CVPR2013 [4]
PCA-ITQ	15.67	16.20	16.64	22.46	25.30	27.09	TPAMI2013 [3]
DH	16.17	16.62	16.96	23.79	26.00	27.70	CVPR2015 [13]
DA	16.82	17.01	17.21	**24.54**	**26.62**	**28.06**	CVPR2016 [6]
DeepBit	**19.43**	**24.86**	**27.73**	–	–	–	CVPR2016 [12]
UH-BDNN	17.83	18.52	–	–	–	–	ECCV2016 [2]
SADH-L	16.45	16.48	17.61	19.11	19.42	19.96	TPAMI2018 [25]
Proposed	16.13	17.04	17.32	21.27	25.23	26.98	–

Table 3: Retrieval performance comparison with state-of-the-art methods on the CUHK03 dataset with manually labeled pedestrian detections. Results are shown using the Cumulative Matching Characteristic (CMC) curve. The numerical suffix of each method denotes the hash code length. First 5 rows shows the performance of supervised approaches training on the same dataset. Last 8 results represent the recognition performance obtained by unsupervised methods. The majority of the results have been taken from [33]. Best result for each rank is in boldface font.

Method / Rank →	1	5	10	20	30	Publication
DRSCH-128	18.74	48.39	**69.66**	**81.03**	**91.28**	TIP2015 [33]
DRSCH-64	21.96	46.66	66.04	78.93	88.76	TIP2015 [33]
DSRH-128	8.08	26.10	45.82	64.95	79.03	CVPR2015 [34]
DSRH-64	14.44	43.38	66.77	79.19	87.45	CVPR2015 [34]
FPNN	**20.65**	**50.09**	66.42	80.02	87.71	CVPR2014 [10]
KSH-CNN-128	3.65	11.71	19.75	30.68	43.46	CVPR2012 [15]
KSH-CNN-64	3.12	12.90	19.96	32.59	45.62	CVPR2012 [15]
MLH-CNN-128	2.75	11.62	24.61	39.68	49.26	ICML2011 [23]
MLH-CNN-64	1.75	8.14	19.60	35.64	47.45	ICML2011 [23]
BRE-CNN-128	3.91	7.24	11.83	24.20	36.15	NIPS2009 [9]
BRE-CNN-64	3.22	6.74	10.25	24.69	37.75	NIPS2009 [9]
Proposed-64	**7.21**	**15.19**	**26.10**	**39.97**	**49.59**	–
Proposed-16	3.85	10.52	18.59	28.07	37.19	–

[5] Jae Pil Heo, Youngwoon Lee, Junfeng He, Shih Fu Chang, and Sung Eui Yoon. 2012. Spherical hashing. In *CVPR*. 2957–2964.

[6] Chen Huang, Chen Change Loy, and Xiaoou Tang. 2016. Unsupervised Learning of Discriminative Attributes and Visual Representations. In *CVPR*. 5175–5184.

[7] Piotr Indyk and Alexandr Andoni. 2006. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM* 51, 1 (2006), 117–122.

[8] Alex Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. Department of Computer Science, University of Toronto, Toronto, ON, Canada. 1—-60 pages.

[9] Brian Kulis and Trevor Darrell. 2009. Learning to hash with binary reconstructive embeddings. In *NIPS*. 1–9.

[10] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. 2014. DeepReID: Deep Filter Pairing Neural Network for Person Re-identification. In *CVPR*. Ieee, 152–159.

[11] Guosheng Lin, Chunhua Shen, David Suter, and Anton Van Den Hengel. 2013. A general two-step approach to learning-based hashing. In *ICCV*. 2552–2559.

[12] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. 2016. Learning Compact Binary Descriptors with Unsupervised Deep Neural Networks. In *CVPR*. 1183–1192.

[13] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep hashing for compact binary codes learning. In *CVPR*, Vol. 07-12-June. 2475–2483.

[14] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2016. Deep Supervised Hashing for Fast Image Retrieval. In *CVPR*. 2064–2072.

[15] Wei Liu, Jun Wang, Rongrong Ji, Yu Gang Jiang, and Shih Fu Chang. [n. d.]. In *CVPR*.

[16] Xianglong Liu, Cheng Deng, Bo Lang, Dacheng Tao, and Xuelong Li. 2016. Query-Adaptive Reciprocal Hash Tables for Nearest Neighbor Search. *IEEE TIP* 25, 2 (2016), 907–919.

[17] Yang Long, Li Liu, Fumin Shen, Ling Shao, and Xuelong Li. 2017. Zero-shot Learning Using Synthesised Unseen Visual Data with Diffusion Regularisation. *IEEE TPAMI* 8828, c (2017), 1–14.

[18] Niki Martinel, Matteo Dunnhofer, Gian Luca Foresti, and Christian Micheloni. 2017. Person Re-Identification via Unsupervised Transfer of Learned Visual Representations. In *ICDSC*. 1–6.

[19] Niki Martinel and Christian Micheloni. 2014. Sparse Matching of Random Patches for Person Re-Identification. In *ICDSC*. 1–6.

[20] Niki Martinel, Christian Micheloni, and Gian Luca Foresti. 2015. The Evolution of Neural Learning Systems: A Novel Architecture Combining the Strengths of NTs, CNNs, and ELMs. *IEEE SMC Magazine* 1, 3 (jul 2015), 17–26.

[21] Niki Martinel, Christian Micheloni, and Claudio Piciarelli. 2013. Learning pairwise feature dissimilarities for person re-identification. In *ICDSC*. 1–6.

[22] Niki Martinel, Claudio Piciarelli, Gian Luca Foresti, and Christian Micheloni. 2016. Mobile Food Recognition with an Extreme Deep Tree. In *ICDSC*. 56—-61.

[23] Mohammad Norouzi, Dm Blei, and David Fleet. 2011. Minimal Loss Hashing for Compact Binary Codes. In *ICML*. 353–360.

[24] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *IJAR* 50, 7 (2009), 969–978.

[25] Fumin Shen, Yan Xu, Li Liu, Yang Yang, Zi Huang, and Heng Tao Shen. 2018. Unsupervised Deep Hashing with Similarity-Adaptive and Discrete Optimization. *IEEE TPAMI* 8828, c (2018), 1–1.

[26] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR* (2015), 1–14.

[27] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. 2017. Deep Hashing Network for Unsupervised Domain Adaptation. In *CVPR*.

[28] Jun Wang, Sanjiv Kumar, and Shih Fu Chang. 2010. Semi-supervised hashing for scalable image retrieval. In *CVPR*. 3424–3431.

[29] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. 2014. Hashing for Similarity Search: A Survey. *arXiv preprint* (2014), 1–29. arXiv:1408.2927

[30] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. 2017. A Survey on Learning to Hash. *IEEE TPAMI* 13, 9 (2017). arXiv:1606.00185

[31] Y Weiss, A Torralba, R Fergus Advances in neural Information, and Undefined 2009. 2008. Spectral hashing. In *NIPS*. 1–8.

[32] Haofeng Zhang, Li Liu, Yang Long, and Ling Shao. 2018. Unsupervised Deep Hashing With Pseudo Labels for Scalable Image Retrieval. *IEEE TIP* 27, 4 (apr 2018), 1626–1638.

[33] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang. 2015. Bit-Scalable Deep Hashing With Regularized Similarity Learning for Image Retrieval and Person Re-Identification. *IEEE TIP* 24, 12 (dec 2015), 4766–4779.

[34] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. 2015. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*. 1556–1564.

[35] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. 2016. Deep Hashing Network for Efficient Similarity Retrieval. In *AAAI Conference on Artificial Intelligence*. 2415–2421.