Hierarchical Recurrent Neural Hashing for Image Retrieval With Hierarchical Convolutional Features

Xiaoqiang Lu, *Senior Member, IEEE*, Yaxiong Chen, and Xuelong Li, *Fellow, IEEE*

***Abstract*— Hashing has been an important and effective technology in image retrieval due to its computational efficiency and fast search speed. The traditional hashing methods usually learn hash functions to obtain binary codes by exploiting hand-crafted features, which cannot optimally represent the information of the sample. Recently, deep learning methods can achieve better performance, since deep learning architectures can learn more effective image representation features. However, these methods only use semantic features to generate hash codes by shallow projection but ignore texture details. In this paper, we proposed a novel hashing method, namely hierarchical recurrent neural hashing (HRNH), to exploit hierarchical recurrent neural network to generate effective hash codes. There are three contributions of this paper. First, a deep hashing method is proposed to extensively exploit both spatial details and semantic information, in which, we leverage hierarchical convolutional features to construct image pyramid representation. Second, our proposed deep network can exploit directly convolutional feature maps as input to preserve the spatial structure of convolutional feature maps. Finally, we propose a new loss function that considers the quantization error of binarizing the continuous embeddings into the discrete binary codes, and simultaneously maintains the semantic similarity and balanceable property of hash codes. Experimental results on four widely used data sets demonstrate that the proposed HRNH can achieve superior performance over other state-of-the-art hashing methods.**

***Index Terms*— Image retrieval, supervised hashing, hierarchical convolutional features, hierarchical RNN**

I. Introduction

RECENT years have witnessed the explosive growth of data in the internet, and approximate nearest neighbors (ANN) search [1] has attracted increasing attention in image retrieval, which tries to seek the approximate nearest neighbors of the query from a large scale datasets. Among existing ANN search technologies, hashing has become an important and effective technology due to its computational efficiency and fast search speed [2]–[4].

Manuscript received February 17, 2017; revised June 30, 2017 and August 22, 2017; accepted September 12, 2017. Date of publication September 22, 2017; date of current version October 17, 2017. This work was supportedin part by the National Natural Science Foundation of China under Grant 61761130079, Grant 61472413, and Grant 61772510, in part by the Key Research Program of Frontier Sciences, CAS, under Grant QYZDY-SSW-JSC044, and in part by the Young Top-notch Talent Program of Chinese Academy of Sciences under Grant QYZDB-SSW-JSC015. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Guoliang Fan. *(Corresponding author: Xiaoqiang Lu.)*

The authors are with the Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: luxq666666@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2755766

In the early years, the principal focus of research work has been on the data-independent hashing methods, which apply random projection to learn hash function without any training data. The typical representatives are *locality-sensitive hashing* (LSH) [2] and its extensions [5], [6]. However, these methods usually require long hash codes to achieve good retrieval results, which requires large amount of storage costs [7]–[9].

To produce more effective hash codes, researchers proposed data-dependent hashing methods, which use the training data to learn hashing functions. Data-dependent hashing methods are mainly divided into unsupervised methods and supervised methods. Unsupervised methods, including *iterative quantization* (ITQ) [3] and *spectral hashing* (SH) [7], only use unlabeled training data to generate efficient hash codes. To preserve hash codes semantic similarity, supervised methods, including *boosted similarity sensitive coding* (BCCS) [10], *hamming distance metric learning* (HDML) [11], and *ranking-based supervised hashing* (RSH) [12], are explored to use label information. But those methods apply linear projections to learn similarity-preserving hash functions, they cannot well handle the nonlinear structure of data. To address this problem, *binary reconstructive embedding* (BRE) [13] and *supervised hashing with kernels* (KSH) [14] are presented to learn nonlinear hash functions in kernel space. For the vast majority of unsupervised and supervised methods, each input sample is represented by a hand-crafted feature vector (e.g., GIST [15], SIFT [16]), which cannot optimally represent the information of the sample and thus limit the retrieval performance in practice [3], [17], [18].

To overcome this limitation, most recently, many methods [8], [9], [19], [20] have exploited *convolutional neural work* (CNN) to learn more effective image representation and obtained better performance over the conventional hash methods. However, these methods still exist the following challenges. Firstly, they often use sum-pooling or max-pooling operation to deal with convolutional feature maps [19], [21], [22]. The pooling operation in essence is to transform a matrix into a point, so it fail to consider the spatial structure of convolutional feature maps, which will lead to substantial information loss and eventually affect the retrieval performance [23]. Secondly, these deep hashing methods exploit semantic features to generate hash codes by shallow projection but ignore spatial details [8], [20], which is also critical to generate effective hash codes [24], as illustrated in Fig.1(a). Finally, there exists uncontrollable quantization error in the learned process of deep hashing, which is not optimally compatible with binarizing the continuous values into the

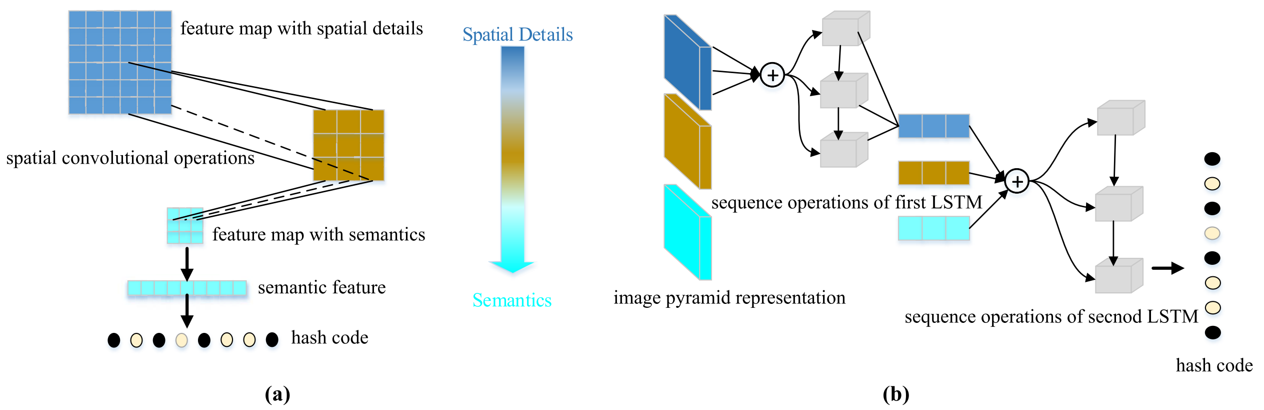
feature map with spatial details
spatial convolutional operations
feature map with semantics
semantic feature
hash code
Spatial Details
Semantics
sequence operations of first LSTM
image pyramid representation
sequence operations of secnod LSTM
hash code
(a)
(b)

Fig. 1. A comparison between spatial convolutional operations of DSRH for hashing and sequence operations of HRNH. DSRH leverages spatial convolutional operations to obtain semantic features, which is further exploited to generate hash codes by shallow projection but ignore spatial details. In HRNH, Hierarchical RNN is exploited to learn hash functions by using image pyramid representation that contains spatial details and semantic information simultaneously. (a) Spatial convolutional operations of DSRH for Hashing. (b) Sequence operations of Hierarchical Recurrent Neural Hashing.

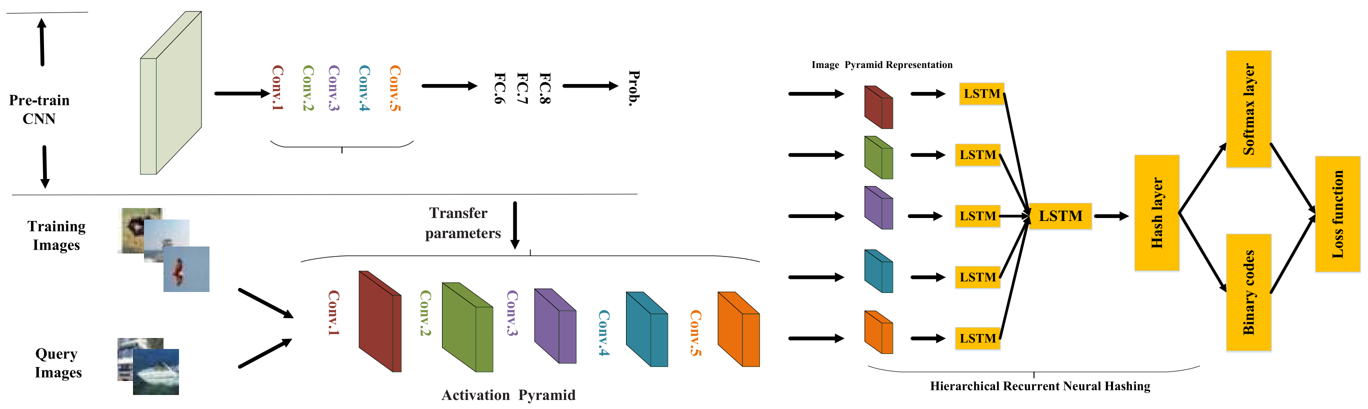

Fig. 2. Overview of the proposed deep learning framework. Firstly, we extract convolutional feature maps from the *conv1, conv2, conv3, conv4* and *conv5* convolutional layer by using the VGG-F pre-trained on ImageNet. And then we adopt the bilinear interpolation and the similarity selection strategy on feature maps of each convolutional layers to obtain image pyramid representation. The deep network structure of HRNH consists of two LSTM layers, an hash layer, and an softmax layer. The number of LSTM hidden units is set to 128. And the hash layer has K (the code length) nodes. Finally, the loss function is designed to maintain the semantic similarity and balanceable property of hash codes, and simultaneously consider the quantization error of binarizing the continuous embeddings into the discrete binary codes.

discrete binary codes and ultimately affects the quality of binary codes [8], [9].

In this paper, we propose a novel *Hierarchical Recurrent Neural Hashing* (HRNH) method, which leverages hierarchical *recurrent neural network* (RNN) to learn discriminative binary codes, as shown in Fig.2. To leverage the spatial structure of the convolutional feature map, LSTM is introduced to learn feature vector by using the convolutional feature map as input. Each column of the convolutional feature map is input into LSTM in a fixed order, which is the arrayed order of the column of the convolutional feature map. In order to deal with a sequence of convolutional feature maps, hierarchical LSTM is introduced to learn hash codes by directly leveraging the convolutional feature maps as input. The detailed process is as follows: each feature map is input into the first LSTM layer to obtain a proper abstract of each feature map, which is the last hidden state of the first LSTM and one-dimensional feature vector. After inputting many feature maps of an image into the first LSTM layer, we can obtain a sequence of feature vectors. A sequence of feature vectors is then fed into the second LSTM layer. Two LSTM layers are trained in end-to-end way. Thus, compared to the sum-pooling or max-pooling operation with convolutional feature maps, the proposed method can

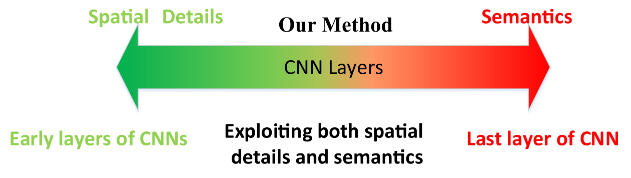

Fig. 3. Convolutional layers of a representative CNN model. The early convolutional layer tends to retain spatial details. On the other hand, the last convolutional layer can capture more semantics and less spatial details.

leverage the spatial structure of convolutional feature maps. Furthermore, considering spatial details and semantic features simultaneously, image pyramid representation is constructed by hierarchical convolutional features that are extracted from multiple convolutional layers of pre-trained CNN, as shown in Fig.1(b). The early convolutional layer tends to retain spatial details asymptotically while the last convolutional layer can capture more semantics and less spatial details [25], as illustrated in Fig.3. Hence, image pyramid representation contains spatial details and semantic information simultaneously. But the size and the number of feature maps form different convolutional layers are different. In this case, we adopt the bilinear interpolation and the similarity selection strategy to obtain the same size and number of feature maps form different convolutional layers. And constructing the same size and number of feature maps form different convolutional layers can help hierarchical RNN to generate efficient hash codes (see Section IV-D). Hence, the proposed method exploits spatial details and semantic features to generate hash codes by using deep underlying structure, which is hierarchical RNN. Finally, a new loss function is designed to control the quantization error of binarizing the continuous embeddings into the discrete binary codes, and simultaneously maintain the semantic similarity and balanceable property of hash codes. Experimental results demonstrate that the proposed HRNH can achieve superior performance over other state-of-the-art hashing methods.

The main contributions of our paper are in the following several aspects:

1) We propose a novel deep learning framework by leveraging hierarchical RNN to generate effective hash codes. And hierarchical RNN can exploit directly convolutional feature maps as input to preserve the spatial structure of convolutional feature maps, which can get more effective information to generate hash codes.
2) We propose a novel hashing method to learn similarity-preserving hash functions for images by using image pyramid representation that contains spatial details and semantic information simultaneously. And image pyramid representation is composed of hierarchical convolutional features that are extracted from multiple convolutional layers of pre-trained CNN.
3) We propose a new loss function that maintains the semantic similarity and balanceable property of hash codes, and simultaneously considers the quantization error of binarizing the continuous embeddings into the discrete binary codes.

The remainder of this paper is structured as follows. Section II elaborates a brief review of the literature about image retrieval. Section III presents the procedure of our HRNH model. Section IV shows the details and results of the experiment. Section V introduces conclusions about our proposed HRNH.

II. Related Works

We will make a brief review of the literature from several related aspects, *i.e.*, hierarchical recurrent neural networks, aggregating convolutional features for retrieval, hashing methods without using deep neural network, and deep hashing methods.

A. Hierarchical Recurrent Neural Networks

Hierarchical RNN has been extensively used in many visual fields. For example, Pan *et. al* [23] introduced a new video captioning method by leveraging hierarchical recurrent encoder to model temporal information of videos. Yu *et al.* [26] presented a novel video captioning method that generated one or multiple sentences to depict videos by using hierarchical RNN. Sutskever *et al.* [27] developed a novel end-to-end sequence learning method by exploring hierarchical RNN to tackle an English to French translation task. Sordoni *et al.* [28] proposed a probabilistic suggestion method by exploiting hierarchical recurrent encoder-decoder architecture to generate Context-Aware Query Suggestion. In this paper, hierarchical RNN is applied to learn hash function for image retrieval.

B. Aggregating Convolutional Features for Retrieval

CNNs have been widely applied in many vision tasks due to the impressive learning power [29], [30]. Recent studies [19], [22], [31], [32] show that the activations of the convolutional layers can be exploited as image features, which achieve superior performance for image retrieval. Razavian *et al.* [31] proposed to decompose the samples into multiple patches and extract patch feature representations utilizing CNN. However, they need leverage all the patches to cross-match during searching. Obviously, this method is not suitable for large-scale datasets since matching all the patches will result in high computational cost. Kalantidis *et al.* [32] presented a simple but straightforward retrieval method which creates global representations through cross-dimensional weighting and aggregation of convolutional feature maps. Babenko and Lempitsky [19] introduced a novel aggregation method which leverages sum-pooling to generate compact representations by aggregating convolutional feature maps. But they cannot achieve precise descriptor matching. Tolias *et al.* [22] developed an effective aggregation method which segments the convolutional feature maps into multiple patches at different scales and generates global descriptors to achieve precise descriptor matching by using sum-pooling to aggregate them. And the authors also presented to leverage max-pooling to aggregate convolutional feature maps. Their method has better performance than the method in [19]. Recent works [33]–[35] have summarized the research of image retrieval. Wang *et al.* [33] presented a comprehensive survey about the learning to hash algorithms. Zheng *et al.* [34] developed a new image search method that implemented accurate visual matching by using probabilistic analysis to obtain discriminative cues. Zhang *et. al* [35] provided a comprehensive survey about SIFT-based and CNN-based methods of instance retrieval. Instead of using the pooling operators on feature maps, the proposed method directly uses the entire feature maps as an input of Hierarchical RNN to learn global descriptors.

C. Hashing Methods Without Using Deep Neural Network

Hashing methods have been widely applied in nearest neighbor search since their fast query and space complexity. In the early years, the principal focus of research work has been on the data-independent hashing methods. For example, *locality sensitive hashing* (LSH) [2] is one of the famous data-independent hashing methods, which leverage random projections to generate hashing bits. Jain *et al.* [5] introduced a fast image search method by learning Mahalanobis metrics to capture the underlying relationships of the images. Then learned metric parameterization is encoded as a randomized locality-sensitive hash function. However, the aforementioned methods need long hash codes to achieve good retrieval results, which require large amount of storage costs.

To generate more discriminative hash codes, researchers proposed data-dependent hashing methods, which use the training data to learn hashing functions. Those methods can be mainly categorized into unsupervised methods and supervised methods. Unsupervised methods only use unlabeled training data to generate efficient binary codes. For example, Kulis and Grauman [36] presented a *kernelized locality-sensitive hashing* (KLSH) method by producing locality-sensitive hashing, which can accommodate the similar projection of the kernel function. Weiss *et al.* [7] introduced a *spectral hashing* (SH) method by dealing with the problem of graph partitioning to generate balanced binary codes.

To preserve hash codes semantic similarity, supervised methods are explored to use label information. For example, Shakhnarovich [10] introduced a *boosted similarity sensitive coding* (BCCS) method by using label information in a boosting algorithm to learn a weighted Hamming encoding. Norouzi *et al.* [11] developed a *hamming distance metric learning* (HDML) method by using a flexible form of triplet ranking loss and minimizing a piecewise smooth upper bound on the empirical loss to overcome discontinuous optimization of the discrete mappings. Wang *et al.* [12] presented a *ranking-based supervised hashing* (RSH) method by leveraging a set of rank triplets to learn linear projection-based hash functions. However, those methods apply linear projections to learn similarity-preserving hash functions, they cannot well handle the nonlinear structure of data. To solve this problem, *binary reconstructive embedding* (BRE) [13] and *supervised hashing with kernels* (KSH) [14] are presented to learn nonlinear hash functions. But these methods generate hash codes by using shallow projection, which cannot capture relatively complex semantic information of the sample [37], [38]. Some recent works illustrated that there exist functions that are significantly more efficiently approximated by deeper projection compared with shallower ones [39], [40]. LSTM [41] is a compact and powerful neural network which captures well the nonlinear relationship of the data. This paper is different from these literatures in that we explicitly exploit deep neural network to find multiple hierarchical non-linear projections to produce more discriminative and compact hash codes.

D. Deep Hashing Methods

Deep learning becomes more and more popular in image retrieval, and researchers have been exploiting deep learning to effectively learn image representations and hash functions. For example, Xia *et al.* [42] combined SAE with RBMs to generate discriminative binary codes. Liong *et al.* [43] introduced *deep hashing* (DH) and *supervised deep hashing* (SDH) methods by leveraging nonlinear projections to learn similarity-preserving hash functions. Do *et al.* [44] improved DH and SDH methods by proposing *supervised discrete hashing with deep neural network* (SDH-DNN) and *unsupervised discrete hashing with deep neural network* (UDH-DNN) method, which used the auxiliary variable to overcome binary constraints. And Do *et al.* [45] also presented *supervised hashing with binary deep neural network* (SH-BDNN) and *unsupervised hashing with binary deep neural network* (UH-BDNN) method by constraining one hidden layer to obtain binary codes. Other deep hashing methods leverage CNN to learn similarity-preserving hash function. Liu *et. al* [38] developed a new *deep supervised hashing* (DSH) method to generate similarity-preserving hash codes by using a CNN architecture, which leveraged paired images as inputs. Li *et al.* [46] proposed a *deep pairwise-supervised hashing* (DPSH) method by performing features learning and hash-code learning simultaneously. Lai *et al.* [47] developed a supervised deep hashing method by using deep neural networks to project images into binary codes. Xia *et al.* [20] proposed CNNH that trains a CNN to fit the approximate hash codes calculated from the similarity matrix. Zhang *et al.* [9] improved CNNH by presenting a *deep regularized similarity comparison hashing* (DRSCH) method that imposing the novel regularization term to learn hash functions. Zhao *et al.* [8] presented a *deep semantic ranking hashing* (DSRH) method by considering the ranking constraint to preserve multilevel semantic similarity. Although most deep hashing methods has made progress, they exist a crucial disadvantage that the quantization error is not optimally minimized, thus uncontrollable quantization error intrinsically affects the retrieval performance. The proposed method designs a new loss function that considers the quantization error of binarizing the continuous embeddings into the discrete binary codes. And we will describe the proposed method in next section.

III. The Proposed Method

Let $\boldsymbol{I} = \{\mathcal{I}^{(n)}\}_{n=1}^{N}$ be N images, $\boldsymbol{Y} = \{y^{(n)}\}_{n=1}^{N}$ be their corresponding labels, where $\mathcal{I}^{(n)}$ $(1 \leq n \leq N)$ is the n-th image, $y^{(n)}$ $(1 \leq n \leq N)$ is the n-th label. The purpose of hash learning is to find a proper mapping function $\mathcal{H} : \boldsymbol{X} = \{\mathcal{I}^{(n)}\}_{n=1}^{N} \rightarrow \{-1, 1\}^{K \times N}$ that encodes images to K-bit binary codes while maintaining the semantic structure of images in the original space.

Existing hashing methods adopted only semantic features to generate hash codes but ignored spatial details. In this paper, we propose a novel hashing method called *hierarchical recurrent neural hashing* (HRNH) to learn discriminative binary codes by using image pyramid representations that contains spatial details and semantic information simultaneously.

As shown in Fig.2, the proposed HRNH contains three main components: 1) An image pyramid representations is designed to learn hash functions for images, which contain hierarchical convolutional features that are extracted from multiple convolutional layers of pre-trained CNN. 2) A hierarchical RNN can directly use hierarchical convolutional feature maps as input to generate discriminative binary codes. 3) A new loss function is designed to maintain the semantic similarity and balanceable property of hash codes, and simultaneously consider the quantization error of encoding the continuous hash codes into the discrete binary codes. In the following section, we first describe the composition of hierarchical convolutional features, and then we present Long-Short Term Memory neuron (LSTM). Furthermore, We leverage a Hierarchical LSTM to generate binary codes. Finally, we introduce the new loss function as well as theoretical analysis.

A. Convolutional Features

Recent works [19], [22], [31], [32] show that the activations of the convolutional layers can be used as image features,

TABLE I
CONFIGURATION OF OUR NETWORK

Layers	Configuration
$conv1$	filter 64×11×11, stride 4×4, pad 0, LRN, pool 2×2
$conv2$	filter 256×5×5, stride 1×1, pad 2, LRN, pool 2×2
$conv3$	filter 256×3×3, stride 1×1, pad 1
$conv4$	filter 256×3×3, stride 1×1, pad 1
$conv5$	filter 256×3×3, stride 1×1, pad 1, pool 2×2
$fc6$	4096, dropout
$fc7$	4096, dropout
$fc8$	1000, classification

which achieve superior performance. Thus we extract the activations of multiple convolutional layers through a pre-trained network, e.g., AlexNet [48] or VGG-Net [49], to from image pyramid representation. And the focus of this paper is not to learn different CNN. Hence, we just adopt pre-trained VGG-F [49] to demonstrate the effectiveness of our proposed HRHN. The detail configurations of VGG-F are listed in Table I. The CNN architecture contains five convolutional layers with three pooling operations (*conv* $1-5$), two fully connected layers (*fc* $6-7$) and an Classification layer(*fc* 8). The "filter" parameter specifies the number of convolution kernels; the "stride" parameter denotes the convolution stride; the "pad" parameter denotes the spatial padding; the "pool" denotes whether to exploit the max-pooling operation. the "LRN" denotes whether Local Response Normalization (LRN) [49] is exploited; the "dropout" denotes whether to exploit the regularization by dropout [49].

Given an image $\mathcal{I}$, it is first resized to $n \times n$ to fit the input of the pre-trained network, and then is fed into the pre-trained network in a forward propagation. With the forward propagation, the semantic information is gradually enhanced, as well as a gradual decrement of the spatial details (see Fig.3) [25]. The pre-trained network contains S convolutional layers. Let $\mathcal{C}^s(s=1,2,...,S)$ represents the feature maps of the s-th convolutional layer. Assume that $\mathcal{C}^s$ takes the size of $a_p^s \times d_p^s \times c^s$, where c^s represents the number of feature channels in the s-th convolutional layer. a_p^s and d_p^s represents the width and height of the p-th channel ($p=1,2,...,c^s$), respectively. $\hat{\boldsymbol{x}}_p^s = a_p^s \times d_p^s$ denotes the p-th channel feature map in the s-th convolutional layer. Note that, for a sample with different convolutional layers, the size of feature maps is different. Then, we apply bilinear interpolation on every feature map [25], and the feature vector for the l_1-th location can be given as:

$$\boldsymbol{x}_p^s(l_1) = \sum_{l_2} \boldsymbol{\alpha}_{(l_1,l_2)} \hat{\boldsymbol{x}}_p^s(l_2), \tag{1}$$

where $\boldsymbol{x}_p^s$ denotes the sampled feature map, $\hat{\boldsymbol{x}}_p^s$ denotes the original feature map, $\boldsymbol{\alpha}_{(l_1,l_2)}$ denotes the interpolation weight that hinges on the position of l_1 and l_2 adjacent vectors respectively. Note that the interpolation occurs in spatial space and is regarded as an interpolation of the position [25].

Meanwhile, the number of feature channels is different for different convolutional layers. In this case, we use the following similarity strategy to select the same number of feature channels for different convolutional layers [50]. Firstly,

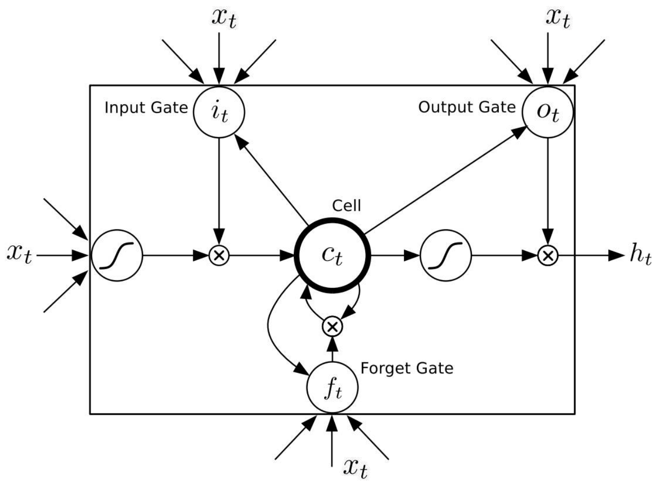

Fig. 4. Long Short-Term Memory block with one cell [51].

we adopt the average operator on the feature maps, *i.e.*,

$$\boldsymbol{x}_{avg}^s = \frac{1}{c^s}\sum_{p=1}^{c^s} \boldsymbol{x}_p^s, \tag{2}$$

where $\boldsymbol{x}_{avg}^s$ represents the average of all feature maps in the s-th convolutional layer. Then, the similarity between the p-th channel feature map and the average of all feature maps can be calculated as follows:

$$score_p^s = sim(\boldsymbol{x}_{avg}^s, \boldsymbol{x}_p^s), \tag{3}$$

where $sim()$ denotes a cosine similarity function, $score_p^s$ denotes the similarity between $\boldsymbol{x}_{avg}^s$ and $\boldsymbol{x}_p^s$. The smaller the value of $score_p^s$ is, the higher the similarity between $\boldsymbol{x}_{avg}^s$ and $\boldsymbol{x}_p^s$ is. Finally, according to the ascending sort of $score_p^s$, we select the top $\hat{p}$ ($1 \le \hat{p} \le c^s$) feature maps for each convolutional layer to form the image pyramid representation $\{(\boldsymbol{x}_1^1, \boldsymbol{x}_2^1, ..., \boldsymbol{x}_{\hat{p}}^1), (\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, ..., \boldsymbol{x}_{\hat{p}}^2), ..., (\boldsymbol{x}_1^S, \boldsymbol{x}_2^S, ..., \boldsymbol{x}_{\hat{p}}^S)\}$. And constructing the image pyramid representation can help hierarchical RNN to generate efficient hash codes (see Section IV-D).

B. Long-Short Term Memory (LSTM)

RNN [41] is a kind of neural network which adopt cyclic connections between its units. It not only adopts nonlinear transition function but also can use its internal memory to process arbitrary sequence of input. LSTM is an extension of RNN by using new units to replace the nonlinear units in conventional RNN. Fig. 4 reports a LSTM unit. The LSTM unit contains a memory cell c and three gates, *i.e.* the forget gate f, the input gate i and the output gate o. The memory cell c preserves past records of the inputs observed up to that time step. The forget gate f controls whether the LSTM will forget the current state, the input gate i is applied to control whether the LSTM will read the current input, the output gate o controls whether the LSTM will output the state. These gates can help LSTM to learn long term dependency information of a sequence.

Given an input sequence $x = \{x_1, x_2, ..., x_t\}$, the forget gate f, the input gate i, the output gate o and the memory cell c are derived as follows:

$$i_t = \tau(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + v_i), \tag{4}$$
$$f_t = \tau(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + v_f), \tag{5}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \phi(W_{xc}x_t + W_{hc}h_{t-1} + v_c), \tag{6}$$
$$o_t = \tau(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + v_o), \tag{7}$$
$$h_t = o_t \odot \phi(c_t), \tag{8}$$

where $\tau(.)$ donates the sigmoid function, $\phi(.)$ donates the hyperbolic tangent function, $\odot$ denotes element-wise product, t represents t step state of LSTM, i_t denotes the input gate values, f_t denotes the forget gate values, o_t is the output gate values, c_t denotes the memory cell activation, h_t denotes the output activation of the hidden layer, W_{x*} denotes the weight matrix between the input and LSTM states, W_{h*} denotes the weight matrix between the hidden states, W_{c*} denotes the weight matrix between the memory cell and gates, v_* denotes the biases vector.

C. Hierarchical LSTM for Hashing

The proposed HRNH is shown in Fig.2. In this paper, we adopt pre-trained CNN to obtain image pyramid representation by extracting hierarchical convolutional features, and then hierarchical RNN leverages hierarchical convolutional features to generate hash codes. And hierarchical RNN consists of two LSTM layers. In the following subsections, we will present how to use hierarchical convolutional features to generate hash codes.

We first obtain a proper abstract of each feature map by inputting each feature map into the first LSTM layer (denoted as $LSTM^{feature\ maps}_{abstract}$). The abstract of each feature map can be obtained by

$$\boldsymbol{h}^s_p = LSTM^{feature\ maps}_{abstract}(\boldsymbol{x}^s_p, \boldsymbol{W}_1, \boldsymbol{v}_1), \tag{9}$$

where $\boldsymbol{h}^s_p$ denotes the last hidden state of the first LSTM, $\boldsymbol{x}^s_p$ denotes one of feature maps in the image pyramid representation. $\boldsymbol{W}_1$ and $\boldsymbol{v}_1$ denote all the weights and all the the biases vectors for the first LSTM, respectively. After inputting image pyramid representation into the first LSTM layer, we can obtain a sequence of feature vectors $\boldsymbol{H}_{abstract} = \{(\boldsymbol{h}^1_1, \boldsymbol{h}^1_2, ..., \boldsymbol{h}^1_{\hat{p}}), (\boldsymbol{h}^2_1, \boldsymbol{h}^2_2, ..., \boldsymbol{h}^2_{\hat{p}}), ..., (\boldsymbol{h}^S_1, \boldsymbol{h}^S_2, ..., \boldsymbol{h}^S_{\hat{p}})\}$. To summarize a sequence of feature vectors, second LSTM layer (denoted as $LSTM^{abstract}_{encode}$) is applied to process this task. The procedure can be given as follows:

$$\boldsymbol{h}_{end} = LSTM^{abstract}_{encode}(\boldsymbol{H}_{abstract}, \boldsymbol{W}_2, \boldsymbol{v}_2), \tag{10}$$

where h_{end} denotes the last hidden state of the second LSTM, $\boldsymbol{W}_2$ and $\boldsymbol{v}_2$ denote all the weights and all the the biases vectors for the second LSTM, respectively.

The hidden layer of the second LSTM is fully-connected by the hash layer followed by a hyperbolic tangent function (see Fig.2). Accordingly, the hash codes can be defined as:

$$\boldsymbol{q} = \phi(\boldsymbol{W_H}^T \boldsymbol{h}_{end} + \boldsymbol{v}_H), \tag{11}$$

where $\boldsymbol{W}_H \in \mathbb{R}^{H \times K}$ denotes the weights in the hash layer, $\boldsymbol{v}_H \in \mathbb{R}^{K \times 1}$ denotes the bias parameter, $\boldsymbol{q} = \{q_1, q_2, ..., q_K\} \in \mathbb{R}^{K \times 1}$ denotes K-bit binary code, $\phi(.)$ denotes the hyperbolic tangent function, defined by $\phi(z) = (1 - exp(-2z))/(1 + exp(-2z))$, where z is a real value. The proposed HRNH projects an image pyramid representation into $[-1, 1]^K$. And the hash codes $\boldsymbol{q} \in [-1, 1]^K$ are continuous values. In order to obtain binary codes, a threshold function is defined as:

$$\boldsymbol{b} = sign(\boldsymbol{q}) = sign(q_k), k = 1, 2, ..., K \tag{12}$$

where $sign(.)$ denotes the element-wise sign function, *i.e.* $sign(x) = 1$ if $x > 0$, otherwise $sign(x) = -1$.

D. Loss Functions

Assuming the binary code $\boldsymbol{b}^{(n)}$ of an image $\mathcal{I}^{(n)}$ is used as the input of the softmax layer, the probability of predicting the label $y^{(n)}$ can be defined as follows:

$$p(y^{(n)} = m|\boldsymbol{b}^{(n)}) = \frac{exp(z_m)}{\sum_{j=1}^{M} exp(z_j)}, m = 1, 2, ..., M \tag{13}$$

where $z_m = \boldsymbol{w}_m^T \boldsymbol{b}^{(n)} + v_m$, and $\boldsymbol{w}_m \in \mathbb{R}^{K \times 1}$ denotes the m-th weight parameter in the softmax layer, v_m denotes the m-th bias parameter in the softmax layer. M denotes the class number of training images. Please note that $\boldsymbol{b}^{(n)} \in \{-1, 1\}^K$.

By considering the negative log-likelihood of the labels $\boldsymbol{Y}$, the following optimization problem can be obtained by:

$$\begin{aligned}\min_{\{\boldsymbol{B}\}} \Im_1 &= -\log p(\boldsymbol{Y}|\boldsymbol{B}) = -\sum_{n=1}^{N} \log p(y^{(n)}|\boldsymbol{B}) \\ &= -\sum_{n=1}^{N}\sum_{m=1}^{M} \mathbf{1}\{y^{(n)} = m\} \log \frac{exp(z_m)}{\sum_{j=1}^{M} exp(z_j)},\end{aligned} \tag{14}$$

where $\mathbf{1}\{.\}$ denotes the indicator function and is 1 if $y^{(n)} = m$ and 0 otherwise, $\boldsymbol{B} = \{b^{(n)}\}_{n=1}^{N}$ denotes the binary codes of all images. The above optimization problem explores label information to preserve hash codes semantic similarity.

Let $b^{(n_1)}$ be the binary code of an image $\mathcal{I}^{(n_1)}$, $b^{(n_2)}$ be the binary code of an image $\mathcal{I}^{(n_2)}$, which is similar to the image $\mathcal{I}^{(n_1)}$. $b^{(n_3)}$ be the binary code of an image $\mathcal{I}^{(n_3)}$, which is dissimilar to the image $\mathcal{I}^{(n_1)}$. $dist_H(b^{(x)}, b^{(y)})$ denotes the Hamming distance between the binary codes $b^{(x)}$ and $b^{(y)}$. We can see that the optimization problem in Eq.14 can effectively make the Hamming distance $dist_H(b^{(n_1)}, b^{(n_2)})$ of two semantically similar images $\mathcal{I}^{(n_1)}$ and $\mathcal{I}^{(n_2)}$ as small as possible, and coinstantaneously make the Hamming distance $dist_H(b^{(n_1)}, b^{(n_3)})$ of two dissimilar images $\mathcal{I}^{(n_1)}$ and $\mathcal{I}^{(n_3)}$ as large as possible [46].

The above optimization problem with binary constraints is difficult to tackle. In this paper, to tackle the discrete optimization problem, we propose a novel strategy that leverages the continuous relaxation to replace the binary constraints. And we apply the continuous hash codes to replace the discrete binary codes. The optimization problem in Eq.14 can be redefined as follows:

$$\begin{aligned}\min_{\{\boldsymbol{B},\boldsymbol{Q}\}} \Im_2 &= -\sum_{n=1}^{N}\sum_{m=1}^{M} \mathbf{1}\{y^{(n)} = m\} \log \frac{exp(\hat{z}_m)}{\sum_{j=1}^{M} exp(\hat{z}_j)} \\ s.t.\ & \boldsymbol{q}^{(n)} = \boldsymbol{b}^{(n)}, n = 1, 2, 3 \ldots, N \\ & \boldsymbol{q}^{(n)} \in \mathbb{R}^{K \times 1}, n = 1, 2, 3 \ldots, N \\ & q_k^{(n)} \in [-1, 1], k = 1, 2, 3 \ldots, K\end{aligned} \tag{15}$$

where $\Im_2$ denotes the cross-entropy loss, $\hat{z}_m = \boldsymbol{w}_m^T \boldsymbol{q}^{(n)} + v_m$, and $\boldsymbol{Q} = \{\boldsymbol{q}^{(n)}\}_{n=1}^N$, $\boldsymbol{q}^{(n)} = \{q_1^{(n)}, q_2^{(n)}, ..., q_K^{(n)}\}$.

However, the continuous relaxation leads to uncontrollable quantization error [52]. Following [38], [53], an regularizer term $\mathcal{L}$ is considered to control the quantization error. We use the L1-norm between the continuous hash codes and the discrete binary codes as the regularizer term, and the detailed analysis is given in Theorem 1. However,optimizing the L1-norm regularizer term alone might cause this situation where the binary codes are all composed of 1, and ultimately affects the retrieval performance. This is because the optimization of L1-norm term can influence the balanceable property of hash codes [54]. To maintain the balanceable property of hash codes , we exploit the balanceable criterion which is defined as the squared sum of the mean value of hash code [7], [44]. The balanceable criterion encourages each bit of a hash code to be -1 or 1 as uniformly as possible [7], [44]. In order to generate good binary codes, the optimization problem can be given as follows:

$$\begin{aligned}\min_{\{\boldsymbol{B},\boldsymbol{Q}\}} \ \Im_3 &= \Im_2 + \gamma \mathcal{L} + \lambda \mathcal{Q} \\ &= -\sum_{n=1}^{N}\sum_{m=1}^{M} \mathbf{1}\{y^{(n)} = m\} \log \frac{exp(\hat{z}_m)}{\sum_{j=1}^{M} exp(\hat{z}_j)} \\ &\quad + \gamma \sum_{n=1}^{N} \| |\boldsymbol{q}^{(n)}| - |\boldsymbol{b}^{(n)}| \|_1 + \lambda \sum_{n=1}^{N} (\rho(\boldsymbol{q}^{(n)}))^2,\end{aligned} \tag{16}$$

where γ denotes the weight parameter to govern the strength of the regularization term. λ denotes the parameter that control the relative importance of the balanceable criterion. $\rho(.)$ denotes the mean operator. $\|.\|_1$ denotes the L1-norm of vector, $|.|$ denotes the absolute value operator. The regularization term $\mathcal{L}$ encourages the learning hash codes $\boldsymbol{q}^{(n)}$ to approximate either 1 or -1. The balanceable criterion $\mathcal{Q}$ guarantees that each bit of a hash code has the same probability of being 1 or -1.

The regularization term $\mathcal{L}$ is hard to compute the derivative. Following [55], we apply a smooth surrogate of the absolute function $|x| = \log \cosh x$, the regularization term $\mathcal{L}$ can be redefined as follows:

$$\mathcal{L} = \sum_{n=1}^{N}\sum_{k=1}^{K} (\log \cosh (|q_k^{(n)}| - |b_k^{(n)}|)), \tag{17}$$

where $q_k^{(n)}$ denote the k-th position of the hash codes $\boldsymbol{q}^{(n)}$, $b_k^{(n)}$ denotes the k-th position of the binary codes $\boldsymbol{b}^{(n)}$.

By considering Eq.16 and Eq.17, the overall loss function of the proposed HRNH can be defined as follows:

$$\begin{aligned}\min_{\{\boldsymbol{B},\boldsymbol{Q},\boldsymbol{W},\boldsymbol{V}\}} \Im &= -\sum_{n=1}^{N}\sum_{m=1}^{M} \mathbf{1}\{y^{(n)} = m\} \log \frac{exp(\hat{z}_m)}{\sum_{j=1}^{M} exp(\hat{z}_j)} \\ &\quad + \gamma \sum_{n=1}^{N}\sum_{k=1}^{K} (\log \cosh (|q_k^{(n)}| - |b_k^{(n)}|)) \\ &\quad + \lambda \sum_{n=1}^{N} (\frac{1}{K} \sum_{k=1}^{K} q_k^{(n)})^2,\end{aligned} \tag{18}$$

where $\boldsymbol{W}$ and $\boldsymbol{V}$ denotes all the weights and all the the biases vectors for our HRNH model, respectively. The loss function in Eq.18 can be optimized by RMSProp [56]. During the optimal process, the loss function can maintain the semantic similarity and balanceable property of hash codes, and simultaneously control the quantization error of encoding the continuous hash codes into the discrete binary codes.

E. Theoretical Analysis

Inspired by Iterative Quantization (ITQ) [3] that takes into account the quantization loss in the iterative process, we impose the point-wise quantization loss in our deep network. And then we introduce the relationship between our work and ITQ.

In ITQ, the formula of the quantization loss can be given as follows:

$$Q_{ITQ} = \|\boldsymbol{q}^{(n)} - sgn(\boldsymbol{q}^{(n)})\|_2, \tag{19}$$

where Q_{ITQ} denotes the quantization loss, $sgn(.)$ denotes the element-wise sign function, *i.e.* $sgn(x) = 1$ if $x > 0$, otherwise $sgn(x) = -1$.

Theorem 1 (Upper Bound): The proposed regularizer term $\mathcal{L}$ in Eq.16 is an upper bound of the quantization loss in Eq.19.

$$\|\boldsymbol{q}^{(n)} - sgn(\boldsymbol{q}^{(n)})\|_2 \leq \| |\boldsymbol{q}^{(n)}| - |\boldsymbol{b}^{(n)}| \|_1 \tag{20}$$

Proof:

$$\begin{aligned}\|\boldsymbol{q}^{(n)} - sgn(\boldsymbol{q}^{(n)})\|_2 &= \| |\boldsymbol{q}^{(n)}| - |sgn(\boldsymbol{q}^{(n)})| \|_2 \\ &= \| |\boldsymbol{q}^{(n)}| - |\boldsymbol{b}^{(n)}| \|_2 \\ &\leq \| |\boldsymbol{q}^{(n)}| - |\boldsymbol{b}^{(n)}| \|_1.\end{aligned} \tag{21}$$

Theorem 1 indicates that the proposed regularizer term $\mathcal{L}$ in Eq.16 is a reasonable standard for hash-code learning. The regularizer term $\mathcal{L}$ is easily optimized by the back-propagation algorithm because it does not have an alternate optimization procedures as ITQ [3]. Another advantage of the proposed regularizer term is that L1-norm require less computational cost [38] and encourage sparsity [53], that is, the training process can be favorably accelerated and more hash bits can be encouraged to -1 or 1 compared with L2-norm, generating more effective hash codes.

IV. Experiments

A. Dataset

MNIST Dataset[1] [57] contains 70 000 handwritten digits images from "0" to "9". It contains 10 classes, each class has 7,000 grayscale images, and the size of each image is 28×28. Following [9], [58], we randomly sample 10 000 digit images (1 000 images per class) as a test query set and other 60 000 images as the training set.

CIFAR-10 Dataset[2] [59] consists of 60 000 color images collected from 80-million tiny images, it contains 10 objects classes, each class contains 6 000 images, and each image has 32×32 pixels. Following the strategy in [9] and [58], we randomly sample 10 000 images (1 000 images per class) as a test query set and other 50 000 images as the training set.

[1] http://yann.lecun.com/exdb/mnist

[2] http://www.cs.toronto.edu/∼kriz/cifar.html

CIFAR-20 Dataset[3] [59] contains 60 000 color images in 20 superclasses, each class contains 3,000 images, each image has 32×32 pixels. Following the setting in [9] and [58], we randomly sample 500 images per class as a test query set and other 2500 images per class as the training set.

YouTube Faces[4] [60] consists of 621 126 face images for 1,595 different people. And the size of each face category is different. Following the strategy in [61], we randomly choose 100 samples from each of the 65 largest face categories as a test query set. For the unsupervised methods, we adopt all the remaining face samples as the training set. To implement supervised learning, we randomly choose 1,000 samples from each of the 65 largest face categories as the training set.

SUN397[5] [62] contains more than 108 754 images for 397 scene categories. And different categories have different numbers of samples. Following the setting in [61], 100 samples from each of the 18 largest scene categories are randomly chosen to compose a query set. For the unsupervised methods, all the remaining scene samples are adopted for training. For the supervised methods, 200 images from each of the 18 largest scene categories are randomly draw to compose a training set of 3600 scene images, 50 images from each of the 18 largest scene categories are randomly selected to compose a validation set. All the remaining scene samples are used for the database samples for retrieval.

For traditional hashing methods which adopt low-level feature, we use 784 dimensional vector (*i.e.*,28×28) to represent each sample in MNIST, and we apply 512-dimensional GIST feature descriptor [14] to represent each sample in CIFAR-10 and CIFAR-20 and exploit 1,770-dimensional LBP feature vector to represent each face sample in YouTube Faces. For traditional hashing methods which use deep feature representation, we represent each sample in MNIST, CIFAR-10 and CIFAR-20 datasets by 4096-dimensional CNN feature [48]. Meanwhile, we leverage a 1600-dimensional feature vector to represent each scene sample in SUN397, and a 1600-dimensional feature vector is extracted on 12 288-dimensional deep convolutional activation features by using principle component analysis [61]. For deep hashing methods, we apply 28×28 samples as the input in MNIST and use 32×32 samples as the input in CIFAR-10, CIFAR-20. For the proposed method, we first resize all samples of four datasets to 224×224, and then use raw samples as the input.

B. Evaluation Protocols

We adopt three widely used metrics in the literature to evaluate the performance of hashing methods.

- Mean average precision (MAP): We firstly rank all samples by using the Hamming distance between the query sample and the database samples. And then we compute the average precision for each query sample according to the ranked list. Finally, we compute the mean of the average precision.
- Precision@k: We compute the percentage of true neighbors in top k of the ranked list. When k is 500, Precision@500 denotes the percentage of true neighbors in top 500 of the ranked list.
- Precision within Hamming radius 2 (HAM2): It is calculated as the percentage of true neighbors when the Hamming distance between the query sample and the database samples is smaller than 2.

These three metrics reflect the performance of hashing methods in different aspects. If the values of these metrics are higher, the performance of hashing methods is better.

```
Algorithm 1 HRNH algorithm
Input:
    Training images $\boldsymbol{I} = \{\mathcal{I}^{(n)}\}_{n=1}^{N}$ and their corresponding
    labels $\boldsymbol{Y} = \{\boldsymbol{y}^{(n)}\}_{n=1}^{N}$ .
Output:
    All weight parameters $\boldsymbol{W}$ of hierarchical RNN
    All bias parameters $\boldsymbol{V}$ of hierarchical RNN .
Initialization:
    The initial weight between the input and LSTM
    states adopts orthogonal distribution, and the remaining
    weights of hierarchical RNN are randomly sampled by
    glorot_uniform distribution.
Repeat:
1: Calculate the feature map $\hat{\boldsymbol{x}}_p^s$ by forward propagation of
   Pre-train CNN;
2: Use the bilinear interpolation and the similarity selection
   strategy to obtain image pyramid representation
   $\{(\boldsymbol{x}_1^1, \boldsymbol{x}_2^1, ..., \boldsymbol{x}_{\hat{p}}^1), (\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, ..., \boldsymbol{x}_{\hat{p}}^2), ..., (\boldsymbol{x}_1^S, \boldsymbol{x}_2^S, ..., \boldsymbol{x}_{\hat{p}}^S)\}$
   according to (1), (2) and (3);
3: Calculate hash code $\boldsymbol{q}$ and binary code $\boldsymbol{b}$ according to (9),
   (10), (11) and (12);
4: Leverage $\boldsymbol{q}$, $\boldsymbol{b}$ and $\boldsymbol{y}$ to calculate $\Im$ according to (18);
5: Update the parameters $\boldsymbol{W}$ and $\boldsymbol{V}$ by utilizing RMSProp;
Until: a fixed number of iterations
Return: $\boldsymbol{W}$, $\boldsymbol{V}$
```

C. Implementation Detail

We describe the main procedure of the proposed hashing method HRNH in Algorithm 1. The implementation details is presented as follows:

1) Feature Extraction: We use the VGG-F [63] pre-trained on ImageNet [64] by MatConvNet[6] [65] implementation and extract convolutional feature maps from the *conv1, conv2, conv3, conv4* and *conv5* convolutional layer. And then we leverage the bilinear interpolation and the similarity selection strategy to obtain the same number and the same size of feature maps as image pyramid representations. In this paper, according to Section IV-D, the size of each feature map is set to 6×6, and we select the top 24 feature maps for each convolutional layer to from image pyramid representation.

2) Network parameters: The proposed HRNH can be implemented by leveraging the open-source KERAS[7]

[3]http://www.cs.toronto.edu/∼kriz/cifar.html
[4]http://www.cs.tau.ac.il/∼wolf/ytfaces/
[5]http://vision.cs.princeton.edu/projects/2010/SUN/
[6]http://www.vlfeat.org/matconvnet/
[7]https://github.com/fchollet/keras

TABLE II
IMAGE RETRIEVAL RESULTS WITH DIFFERENT HIERARCHICAL RNN ON THE CIFAR-10 DATASET

hierarchical RNN	MAP	precision@500	HAM2
Simple RNN+Simple RNN	38.27	45.43	40.69
Simple RNN+GRU	48.24	57.92	51.15
Simple RNN+LSTM	50.32	59.35	53.58
GRU+GRU	61.02	65.70	62.35
GRU+LSTM	64.12	66.57	64.43
LSTM+LSTM	66.76	67.39	66.94

framework [66]. All the experiments are achieved in workstation with GeForce GTX Titan X GPU, Inter Core i7$-$5930K 3.50GHZ CPU and 64G RAM. We apply RMSProp [56] to optimize the objective function in training stage and we adopt the learning rate 10^{-3}, momentum coefficient 0.9 as defaulted in [67]. The batch size of images is set to 64. The number of LSTM hidden units is set to 128. To generate {8,16, 24, 36, 48, 64}-bit hash codes, the number of units K in the hash layer is set from 8 to 64 respectively. The initial weight between the input and LSTM states adopts orthogonal distribution, the initial weight between the hidden states uses glorot_uniform distribution, the initial weight between the memory cell and gates also uses glorot_uniform distribution. The parameters γ and λ are set as 10^{-4} and 10^{-4}, respectively. The model is trained for 100 epoches, or stopped training until the loss is not decreased on the test set [23].

D. Evaluation of Individual Steps

1) Performance of Different Hierarchical RNN: Hierarchical RNN architectures contain mainly several different RNN modes (*i.e.* Simple RNN, GRU, LSTM) [41]. In this experiment, we compare different hierarchical RNN (*i.e.* Simple RNN+Simple RNN, Simple RNN+GRU, Simple RNN+LSTM, GRU+GRU, GRU+LSTM, LSTM+LSTM) on three metrics for the CIFAR-10 dataset, and image retrieval results with different hierarchical RNN on the CIFAR-10 dateset are presented in Table II. We can observe that hierarchical RNN which contains two LSTM layers can achieve superior performance over other hierarchical RNN architectures on three metrics. Compared with simple RNN, LSTM and GRU have a significant boost for image retrieval. Because the gate of LSTM and GRU can maintain important feature in a long series of steps. Meanwhile, LSTM and GRU contain shortcut paths to bypass multiple time steps. And these shortcut paths reduce the difficulty of vanishing the gradient of error back propagation [68]. As shown in Table II, we can clearly see that LSTM achieves better performance than GRU. This is because GRU computes the new memory content without any control of the amount of the candidate activation. Rather, LSTM unit can possess the amount of the new memory content which can be stored to the memory unit [69]. Thus, LSTM can better leverage image pyramid representation to generate hash code in hierarchical RNN architecture. So we adopt hierarchical RNN which is composed of two LSTM layers in the following experiment.

2) Comparison of the First Layer: RNN has the ability to learn long temporal dependencies on sequential data [41].

TABLE III
IMAGE RETRIEVAL RESULTS WITH THE COMPARISON OF AVERAGE POOLING, FILTER 1×1×1, FILTER1×3×3 AND SIMPLE RNN ON THE CIFAR-10 DATASET

RNN	MAP	precision@500	HAM2
AP+Simple RNN	27.62	32.56	29.13
Filter 1×1×1+Simple RNN	29.38	35.58	31.52
Filter 1×3×3+Simple RNN	30.93	36.24	32.87
Simple RNN+Simple RNN	38.27	45.43	40.69
AP+GRU	35.34	41.28	36.56
Filter 1×1×1+GRU	37.35	33.85	39.72
Filter 1×3×3+GRU	38.26	34.22	41.54
Simple RNN+GRU	48.24	57.92	51.15
AP+LSTM	37.85	43.74	38.94
Filter 1×1×1+LSTM	40.08	47.44	42.76
Filter 1×3×3+LSTM	42.62	48.83	45.15
Simple RNN+LSTM	50.32	59.35	53.58

In this experiment, for the first layer, convolutional layer with filter $1\times1\times1$, convolutional layer with filter $1\times3\times3$, average pooling and simple RNN are applied to deal with the convolutional feature map. Hence, we compare average pooling, filter $1\times1\times1$, filter $1\times3\times3$ and Simple RNN (*i.e.* AP+Simple RNN, Filter $1\times1\times1$+Simple RNN, Filter $1\times3\times3$+Simple RNN, Simple RNN+Simple RNN, AP+GRU, Filter $1\times1\times1$+GRU, Filter $1\times3\times3$+GRU, Simple RNN+GRU, AP+LSTM, Filter $1\times1\times1$+LSTM, Filter $1\times3\times3$+LSTM, Simple RNN+LSTM) on three metrics for the CIFAR-10 dataset, and image retrieval results with the comparison of average pooling, filter $1\times1\times1$, filter $1 \times 3 \times 3$ and RNN on the CIFAR-10 dataset are presented in Table III. It can be shown in Table III that hierarchical RNN containing two RNN layers can achieve superior performance over those methods that use convolutional layer with filter $1\times1\times1$, convolutional layer with filter $1\times3\times3$, and average pooling with the convolutional feature map on three metrics. Because with the sequence of feature maps increases, feature maps contain more semantics and less spatial details in image pyramid representation. What's more, LSTM [41] is a kind of neural network which can use its internal memory to process sequence modeling. It is suitable for dealing with the process of sequence variation. With the sequence of LSTM increases, the feature of the hidden layer also becomes more semantics [68]. Hence, hierarchical RNN can leverage image pyramid representation to generate better hash codes, and the experiment also illustrates the importance of the temporal dependencies among channel feature maps.

3) Performance of Combination: We then evaluate convolutional feature maps from different convolutional layer (*i.e. Original Conv1, Original Conv2, Original Conv3, Original Conv4* and *Original Conv5*), all convolutional feature maps from five convolutional layer (*i.e. All Conv1-5*) and image pyramid representation (*i.e. Top $\hat{p}$ Conv1-5*) on three metrics for the CIFAR-10 dataset, and image retrieval results for convolutional feature maps on the CIFAR-10 dataset are presented in Table IV. We can observe that the proposed method using image pyramid representation can achieve better retrieval

TABLE IV
IMAGE RETRIEVAL RESULTS FOR CONVOLUTIONAL FEATURE MAPS ON THE CIFAR-10 DATASET

Combination	MAP	precision@500	HAM2
Original Conv1	44.45	48.13	45.49
Original Conv2	51.72	53.94	52.63
Original Conv3	50.38	52.62	51.74
Original Conv4	57.04	59.72	58.29
Original Conv5	53.82	56.48	54.70
All Conv1-5	62.43	64.75	63.89
Top $\hat{p}$ Conv1-5	66.76	67.39	66.94

TABLE V
IMAGE RETRIEVAL RESULTS WITH DIFFERENT NUMBER OF FEATURE MAPS ON THE CIFAR-10 DATASET

Number	MAP	precision@500	HAM2
8×5	64.27	65.23	64.57
16×5	65.14	66.60	65.42
24×5	68.28	68.77	69.03
32×5	66.76	67.39	66.94
48×5	66.73	67.25	66.68
64×5	64.28	66.36	64.47

accuracy than the identical method using convolutional feature maps from individual convolutional layer on three metrics. This is because image pyramid representation contains richer spatial details and semantic information than convolutional feature maps from individual convolutional layer. And the proposed method using image pyramid representation outperforms the identical method using all convolutional feature maps in all evaluation metrics, which indicates that the proposed method can obtain effective feature representation, thus leading to better retrieval accuracy.

4) Impact of Feature Maps: We examine the influence of feature maps from the following two aspects: the number of feature map and the size of feature map. We select the top $\hat{p}$ feature maps for each convolutional layer to train the proposed HRNH model and vary $\hat{p}$ from 8 to 64 respectively. Pre-trained CNN have five convolutional layer. Hence, image pyramid representation contains a total of {40, 80, 120, 160, 240, 320} feature maps respectively, and image retrieval results with different number of feature maps on the CIFAR-10 dataset are tabulated in Table V. We can see that the proposed HRNH can achieve better retrieval accuracy when $\hat{p}$ is set to 24. Thus we select the top 24 feature maps for each convolutional layer to evaluate the role of size. The size of feature map is set from 4×4 to 14×14 respectively, and image retrieval results with different size of feature maps on the CIFAR-10 dateset are shown in Table VI. We can observe that the proposed model can achieve better retrieval accuracy when the size of feature map is set to 6×6.

E. Comparison with the State-of-the-art

1) Results on MNIST: To measure the retrieval performance of the proposed HRNH, we implement experiments in the MNIST, CIFAR-10 and CIFAR-20 dataset with several state-of-the-art hashing methods, including unsupervised methods *Locality-Sensitive Hashing* (LSH) [2], *Spectral Hashing* (SH) [7], and *Iterative Quantization* (ITQ) [3], supervised methods *Binary Reconstructive Embedding* (BRE) [13], *Minimal Loss Hashing* (MLH) [70], *Kernel-based Supervised Hashing* (KSH) [14], *Deep Semantic Ranking Hashing* (DSRH) [8], *Deep Similarity Comparison Hashing* (DSCH) [9] and *Deep Regularized Similarity Comparison Hashing* (DRSCH) [9]. LSH, SH, ITQ, BRE, MLH and KSH are traditional hashing methods, which use 784 dimensional vector as input to learn hash function. And other three hashing methods(*i.e.*, BRE-CNN, MLH-CNN and KSH-CNN) use 4096-dimensional CNN feature as input to conduct hash function. While DSRH, DSCH, DRSCH and the proposed HRNH use the raw pixels as input to learn hash function.

TABLE VI
IMAGE RETRIEVAL RESULTS WITH DIFFERENT SIZE OF FEATURE MAPS ON THE CIFAR-10 DATASET

Size	MAP	precision@500	HAM2
4×4	68.07	68.26	68.18
5×5	68.19	68.52	68.42
6×6	68.28	68.77	69.03
7×7	67.78	68.29	68.10
8×8	67.61	67.79	67.66
9×9	65.64	66.25	66.08
10×10	66.31	66.85	66.81
11×11	65.27	66.36	66.23
12×12	65.30	65.71	65.81
13×13	64.98	65.76	65.75
14×14	63.98	65.76	65.75

TABLE VII
IMAGE RETRIEVAL RESULTS (MEAN AVERAGE PRECISION) WITH VARIOUS NUMBER OF BITS ON THE MNIST DATASET. THE SCALE OF TEST QUERY SET IS 10K

Method	MNIST(MAP %)				
	16 bits	24 bits	32 bits	48 bits	64 bits
HRNH	**98.15**	**98.23**	**98.35**	**98.47**	**98.66**
DRSCH[9]	96.92	97.37	97.88	97.91	98.09
DSCH[9]	96.51	96.63	97.21	97.48	97.68
DSRH[8]	96.48	96.69	97.21	97.53	97.75
KSH-CNN[14]	83.89	86.67	88.51	89.41	89.67
MLH-CNN[70]	71.03	76.18	78.06	80.66	80.87
BRE-CNN[13]	61.00	64.05	64.11	66.33	67.02
KSH[14]	82.85	86.03	87.37	88.48	88.82
MLH[70]	45.77	62.16	63.07	65.23	66.70
BRE[13]	41.96	57.19	56.52	64.74	66.55
ITQ [3]	34.44	38.99	40.62	43.04	41.76
SH [4]	13.40	14.81	15.28	16.29	17.11
LSH[2]	22.65	21.39	35.56	27.85	37.78

Table VII shows the retrieval MAP results on the MNIST dataset for different code lengths. Fig.5(a), Fig.5(b) and Fig.5(c) report the precision curves within Hamming radius 2 with various number of bits, the precision curves within top 500 retrieved neighbors with various number of bits, the precision curves within different number of top retrieved neighbors with 64 hash bits on the MNIST dataset respectively. We can distinctly observe that: (1) Traditional hashing methods exploiting 4096-dimensional CNN feature achieve

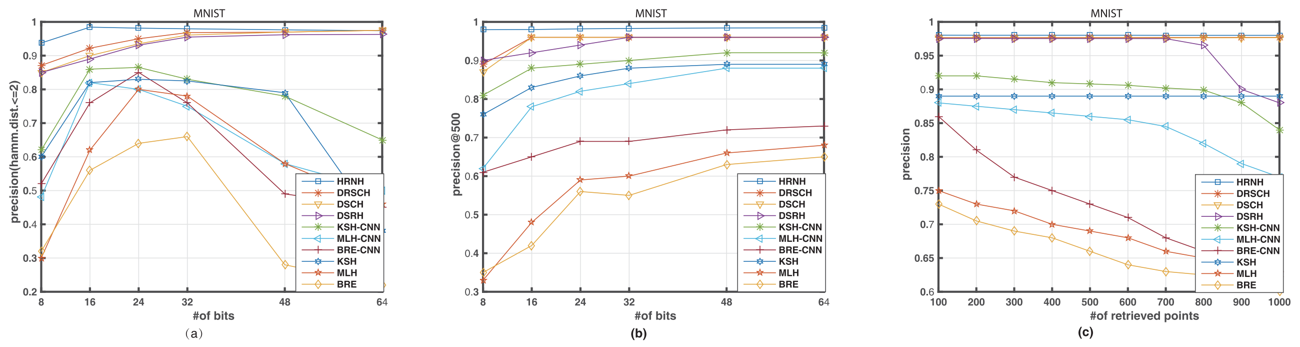

Fig. 5. The results on the MNIST dataset. (a) Precision curves within Hamming radius 2; (b) Precision curves with top 500 returned; (c) Precision curves with 64 hash bits

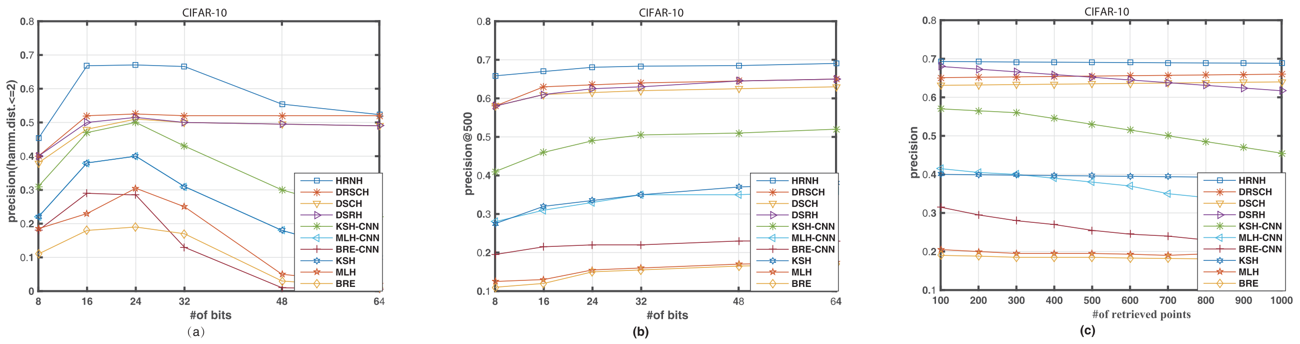

Fig. 6. The results on the CIFAR-10 dataset. (a) Precision curves within Hamming radius 2; (b) Precision curves with top 500 returned; (c) Precision curves with 64 hash bits

superior performance over the identical methods exploiting hand-crafted features, we can find that CNN feature can improve the retrieval accuracy of traditional methods. (2) The proposed HRNH achieves superior performance over other compared state-of-the-art methods for all code lengths.And the proposed HRNH improves the average MAP to 98.37% from 97.55% implemented by DRSCH [9]. (3) From these three figures, although compared state-of-the-art methods have achieved good retrieval accuracy on three widely used metrics, the proposed HRNH still achieves better retrieval accuracy within Hamming radius 2 for all the bits, and better retrieval accuracy within top 500 retrieved neighbors for all the bits, as well as better retrieval accuracy at all retrieved neighbors with 64 hash bits. (4) The proposed HRNH can achieve superior performance over other six supervised hashing methods (*i.e.*, BRE-CNN, MLH-CNN, KSH-CNN,DSRH, DSCH and DRSCH), which illustrates the effectiveness of using image pyramid representations that contains spatial details and semantic information simultaneously.

2) Results on CIFAR-10: Table VIII shows the retrieval MAP results on the CIFAR-10 dataset for different code lengths. We can clearly find that: (1) Comparing traditional methods exploiting hand-crafted feature and the same method exploiting 4096-dimensional CNN feature, we can see that CNN feature can enhance the retrieval accuracy of traditional methods. (2) Compared with other deep hashing methods DSRH [8], DSCH [9], DRSCH [9], DSH [38] and SH-BDNN [45], the proposed HRNH rises the retrieval average MAP from 61.66% (DSRH), 61.65% (DSCH), 62.60% (DRSCH), 65.66% (DSH), 66.18% (SH-BDNN) to 67.72%. This is because the proposed HRNH designs a new loss function, which maintains the semantic similarity and balanceable property of hash codes, and simultaneously considers the quantization error of binarizing the continuous embeddings into the discrete binary codes, thus can rises the retrieval accuracy. (3) The proposed HRNH significantly outperforms other seven supervised hashing methods (*i.e.*, BRE-CNN, MLH-CNN, KSH-CNN,DSRH, DSCH, DRSCH and DSH)

TABLE VIII
IMAGE RETRIEVAL RESULTS (MEAN AVERAGE PRECISION) WITH VARIOUS NUMBER OF BITS ON THE CIFAR-10 DATASET. THE SCALE OF TEST QUERY SET IS 10K

Method	CIFAR-10(MAP %)				
	16 bits	24 bits	32 bits	48 bits	64 bits
HRNH	**65.76**	**67.84**	**67.85**	**68.29**	**68.85**
DSH[36]	64.18	65.12	65.88	67.55	68.06
SH-BDNN[45]	64.30	65.21	65.22	66.22	67.34
DRSCH[9]	61.46	62.19	62.87	63.05	63.26
DSCH[9]	60.87	61.33	61.74	61.98	62.35
DSRH[8]	60.84	61.08	61.74	61.77	62.91
KSH-CNN[14]	40.08	42.98	44.39	45.77	46.56
MLH-CNN[70]	25.04	28.86	31.29	31.88	31.83
BRE-CNN[13]	19.80	20.57	20.59	21.64	21.96
KSH[14]	32.15	35.17	36.51	38.26	39.50
MLH[70]	13.33	15.78	16.29	18.03	18.84
BRE[13]	12.19	15.63	16.10	17.19	17.56
ITQ [3]	11.45	11.63	11.53	10.97	11.24
SH [4]	19.22	19.28	20.09	20.79	21.46
LSH[2]	12.36	11.74	12.30	13.57	12.42

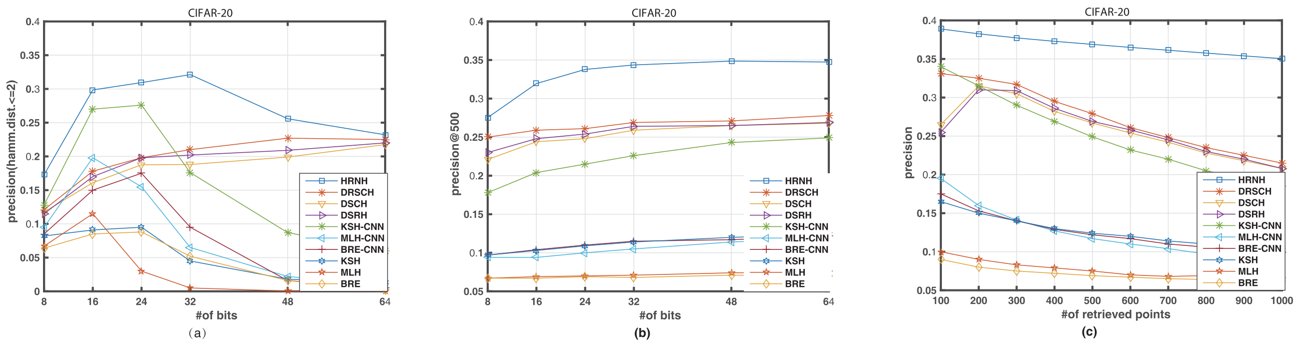

Fig. 7. The results on the CIFAR-20 dataset. (a) Precision curves within Hamming radius 2; (b) Precision curves with top 500 returned; (c) Precision curves with 64 hash bits.

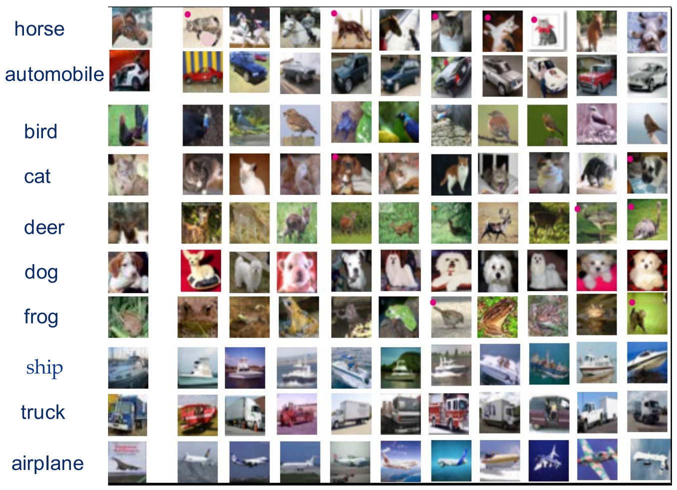

Fig. 8. Top 10 retrieved results from CIFAR-10 dataset by HRNH with 64 bits.The query examples show at the first row, other rows show the retrieval results of HRNH. The false retrieval results are marked with red dot.

TABLE IX

IMAGE RETRIEVAL RESULTS (MEAN AVERAGE PRECISION) WITH VARIOUS NUMBER OF BITS ON THE CIFAR-20 DATASET. THE SCALE OF TEST QUERY SET IS 10K(500 PER CLASS)

Method	CIFAR-20(MAP %)				
	16 bits	24 bits	32 bits	48 bits	64 bits
HRNH	**30.97**	**31.17**	**31.75**	**32.05**	**32.51**
DRSCH[9]	23.41	23.79	24.38	25.63	26.51
DSCH[9]	22.64	23.07	23.88	24.16	24.67
DSRH[8]	22.71	23.39	23.86	24.05	24.74
KSH-CNN[14]	18.53	19.89	21.23	23.11	23.87
MLH-CNN[70]	10.94	12.09	12.89	14.36	15.33
BRE-CNN[13]	9.98	10.67	11.16	11.44	11.95
KSH[14]	9.11	9.42	9.99	10.36	10.92
MLH[70]	7.15	7.32	7.45	7.85	8.10
BRE[13]	7.33	7.62	7.62	8.01	8.11

that leverage semantic information to generate hash codes by shallow projection, because our method uses deep neural network (hierarchical RNN) to generate hash codes by using image pyramid representations that contains spatial details and semantic information simultaneously.

Fig.6(a) illustrates the precision curves within Hamming radius 2 on the CIFAR-10 dataset for various number of bits, we can see that the proposed HRNH achieves best retrieval accuracy within Hamming radius 2 for all the bits. Fig.6(b) reports the precision curves within top 500 retrieved neighbors with various number of bits for the CIFAR-10 dataset, the proposed HRNH consistently perform best retrieval accuracy within top 500 retrieved neighbors for all the bits. Fig.6(c) shows the precision curves within different number of top retrieved neighbors with 64 hash bits on the CIFAR-10 dataset, the proposed HRNH still achieves best retrieval accuracy.

To acquire qualitative visual result, ten query examples from CIFAR-10 dataset by HRNH with 64 bits are showed in Fig.8. Top 10 returned images acquired by different query example are displayed, and the query examples show at the first row, other rows show the retrieval results of HRNH. Note that the false retrieval results are marked with red dot.

3) Results on CIFAR-20: Table IX shows the retrieval MAP results on the CIFAR-20 dataset for different code lengths. The precision curves within Hamming radius 2 on the CIFAR-20 dataset for various number of bits are demonstrated in Fig.7(a). The precision curves within top 500 retrieved neighbors with various number of bits for the CIFAR-20 dataset are reported in Fig.7(b). The precision curves within different number of top retrieved neighbors with 64 hash bits on the CIFAR-20 dataset are illustrated in Fig.7(c). We can clearly see that these retrieval results on three metrics are similar to our observations on the CIFAR-20 dataset. HRNH achieves better performance than other compared state-of-the-art methods, which further illustrates the effectiveness of the proposed method.

4) Results on YouTube Faces: To further verify the retrieval performance of the proposed HRNH in large scale image retrieval, we compare our proposed HRNH on the YouTube Faces dataset with several state-of-the-art hashing algorithms, including unsupervised algorithm LSH [2], SH [7], ITQ [3] and ISOH [71], supervised algorithms *Hamming Distance Metric Learning* (HDML) [11], (CGH) [72], (RSH) [12], Top-RSBC [61], DSCH [9] and DRSCH [9]. Following the identical number of test samples, training samples and identical evaluation metrics as [61], the above eight algorithms use 1,770-dimensional LBP feature vector to represent each face sample. While DSCH, DRSCH and the proposed HRNH use the raw pixels as input to learn hash function. The number of units K in the hash layer is set to 64, 128, and 256, and image retrieval results (mean average precision and percision@100) with 64, 128, and 256 bits on the YouTube Faces dataset

TABLE X

IMAGE RETRIEVAL RESULTS (MEAN AVERAGE PRECISION AND PRECISION@100) WITH 64, 128, AND 256 BITS ON THE YOUTUBE FACES DATASET. THE SCALE OF TEST QUERY SET IS 6500(100 IMAGES FROM EACH OF THE 65 LARGEST FACE CLASSES)

Method	MAP(%)			Precision@100(%)		
	64 bits	128 bits	256 bits	64 bits	128 bits	256 bits
HRNH	**77.26**	**78.04**	**77.83**	**94.16**	**96.54**	**96.61**
DRSCH[9]	72.61	73.34	72.85	90.64	92.38	92.86
DSCH[9]	71.32	72.13	71.69	89.33	91.6	92.05
Top-RSBC[61]	49.14	50.38	52.24	69.6	72.72	72.85
CGH [72]	42.05	47.43	50.12	62.76	69.03	70.9
RSH [12]	30.42	35.87	36.04	58.23	65.36	66.11
HDML [11]	39.03	44.94	47.55	60.66	67.62	69.99
ISOH [71]	16.85	26.64	33.87	34.96	49.67	65.1
ITQ [3]	7.77	24.41	29.76	12.06	42.27	55.04
SH [4]	27.86	34.24	34.22	58.23	64.03	64.61
LSH[2]	1.48	8.37	8.45	4.96	30.97	30.97

TABLE XI

IMAGE RETRIEVAL RESULTS (MEAN AVERAGE PRECISION) WITH VARIOUS NUMBER OF BITS ON THE SUN397 DATASET

Method	SUN397(MAP %)		
	64 bits	128 bits	256 bits
HRNH	**40.42**	**41.08**	**41.06**
DRSCH[9]	37.54	38.23	37.87
DSCH[9]	36.63	37.10	36.74
Top-RSBC[61]	32.30	33.15	34.41
CGH [72]	26.54	28.75	29.82
RSH [12]	11.03	11.32	13.76
HDML [11]	25.64	26.62	28.14
ISOH [71]	11.04	13.09	13.85
ITQ [3]	13.21	15.49	15.81
SH [4]	8.45	9.45	9.68
LSH[2]	1.74	2.51	3.10

are illustrated in Table X. We can observe that the proposed HRNH can achieve superior performance over other compared hashing methods. The proposed HRNH improves the retrieval average MAP to 77.71% from 72.93% implemented by previous best competitor DRSCH [9]. Moreover, the precision within top 100 retrieved neighbors of the proposed HRNH has a significant increase of 3.81% than DRSCH [9]. This is because the proposed HRNH leverages deep network to learn hash function by using image pyramid representation that contains spatial details and semantic information simultaneously.

5) Results on SUN397: To investigate the effectiveness of the proposed HRNH in image search, we further test the proposed HRNH on the SUN397 dataset with several compared state-of-art methods, including unsupervised methods LSH [2], SH [7], ITQ [3] and ISOH [71], supervised methods *Hamming Distance Metric Learning* (HDML) [11], (CGH) [72], (RSH) [12], Top-RSBC [61], DSCH [9] and DRSCH [9]. Following the identical number of test samples, training samples and identical evaluation metrics as [61], the number of units K in the hash layer is set to 64, 128, and 256, and image retrieval results (mean average precision) with 64, 128, and 256 bits on the SUN397 dataset are demonstrated in Table XI. Similar retrieval performance can be seen, the proposed HRNH achieves the retrieval average MAP of 40.85%, which achieves superior performance over other compared hashing methods Top-RSBC (33.28%)DSCH (36.82%) and DRSCH (37.88%). Thus the proposed HRNH outperforms other supervised deep hashing methods, which indicates that the proposed HRNH leverages deep network to learn hash function by using image pyramid representation that contains spatial details and semantic information simultaneously, thus achieving better performance.

V. CONCLUSION

In this paper, we have proposed a novel Hierarchical Recurrent Neural Hashing (HRNH) method, which leverages hierarchical RNN to learn discriminative binary codes by using image pyramid representation for large-scale image retrieval. Firstly, an image pyramid representation is designed to learn similarity-preserving hash functions for images, which is composed of hierarchical convolutional features that are extracted from multiple convolutional layers of pre-train CNN. Secondly, we propose a novel deep learning framework to generate hash codes by leveraging hierarchical RNN, which can take directly convolutional feature maps as input to preserve the spatial structure of convolutional feature maps. Finally, we propose a new loss function, which maintains the semantic similarity and balanceable property of hash codes, and simultaneously considers the quantization error of binarizing the continuous embeddings into the discrete binary codes. Experimental results on the MNIST, CIFAR-10, and CIFAR-20 datasets demonstrate that the proposed HRNH can achieve superior performance over other state-of-the-art hashing methods.

REFERENCES

[1] X. Lu, X. Zheng, and X. Li, "Latent semantic minimal hashing for image retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 355–368, Jan. 2017.

[2] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. VLDB*, 1999, pp. 518–529.

[3] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 817–824.

[4] X. Nie, Y. Chai, J. Liu, J. Sun, and Y. Yin, "Spherical torus-based video hashing for near-duplicate video detection," *Sci. China Inf. Sci.*, vol. 59, p. 059101, May 2016.

[5] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learned metrics," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[6] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1509–1517.

[7] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1753–1760.

[8] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1556–1564.

[9] 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 Process.*, vol. 24, no. 12, pp. 4766–4779, Dec. 2015.

[10] G. Shakhnarovich, "Learning task-specific similarity," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 2005.

[11] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, "Hamming distance metric learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1061–1069.

[12] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang, "Learning hash codes with listwise supervision," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3032–3039.

[13] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1042–1050.

[14] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2074–2081.

[15] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[16] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Sep. 1999, p. 1150.

[17] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, Jul. 2009.

[18] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[19] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proc. ICCV*, 2015, pp. 1269–1277.

[20] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 2156–2162.

[21] L. Zheng, Y. Zhao, S. Wang, J. Wang, and Q. Tian. (2016). "Good practice in CNN feature transfer." [Online]. Available: https://arxiv.org/abs/1604.00133

[22] G. Tolias, R. Sicre, and H. Jégou. (2016). "Particular object retrieval with integral max-pooling of CNN activations." [Online]. Available: https://arxiv.org/abs/1511.05879

[23] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang, "Hierarchical recurrent neural encoder for video representation with application to captioning," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1029–1038.

[24] J. Zhang and Y. Peng. (2016). "SSDH: Semi-supervised deep hashing for large scale image retrieval." [Online]. Available: https://arxiv.org/abs/1607.08477

[25] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 3074–3082.

[26] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu, "Video paragraph captioning using hierarchical recurrent neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4584–4593.

[27] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[28] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J.-Y. Nie, "A hierarchical recurrent encoder-decoder for generative context-aware query suggestion," in *Proc. Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 553–562.

[29] D. Zhang, J. Han, C. Li, J. Wang, and X. Li, "Detection of co-salient objects by looking deep and wide," *Int. J. Comput. Vis.*, vol. 120, no. 2, pp. 215–232, 2016.

[30] D. Zhang, J. Han, J. Han, and L. Shao, "Cosaliency detection based on intrasaliency prior transfer and deep intersaliency mining," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1163–1176, Jun. 2016.

[31] A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Visual instance retrieval with deep convolutional networks," *ITE Trans. Media Technol. Appl.*, vol. 4, no. 3, pp. 251–258, 2016.

[32] Y. Kalantidis, C. Mellina, and S. Osindero, "Cross-dimensional weighting for aggregated deep convolutional features," in *Proc. Eur. Conf. Comput. Vis.*, 2015, pp. 685–701.

[33] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.

[34] L. Zheng, S. Wang, J. Wang, and Q. Tian, "Accurate image search with multi-scale contextual evidences," *Int. J. Comput. Vis.*, vol. 120, no. 1, pp. 1–13, 2016.

[35] L. Zheng, Y. Yang, and Q. Tian, "SIFT meets CNN: A decade survey of instance retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.

[36] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, Jun. 2012.

[37] Z. Zhang, Y. Chen, and V. Saligrama. (2016). "Efficient training of very deep neural networks for supervised hashing." [Online]. Available: https://arxiv.org/abs/1511.04524

[38] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2064–2072.

[39] G. F. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2924–2932.

[40] G. F. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, vol. 39. no. 1, pp. 122–123.

[41] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. (2015). "Character-aware neural language models." [Online]. Available: https://arxiv.org/abs/1508.06615

[42] Z. Xia, X. Feng, J. Peng, and A. Hadid. (2016). "Unsupervised deep hashing for large-scale visual search." [Online]. Available: https://arxiv.org/abs/1602.00206

[43] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2475–2483.

[44] T.-T. Do, A.-Z. Doan, and N.-M. Cheung. (2015). "Discrete hashing with deep neural network." [Online]. Available: https://arxiv.org/abs/1508.07148

[45] T.-T. Do, A.-D. Doan, and N.-M. Cheung, "Learning to hash with binary deep neural network," in *Proc. ECCV*, 2016, pp. 219–234.

[46] W. J. Li, S. Wang, and W. C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 1711–1717.

[47] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3270–3278.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–14.

[50] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1385–1392.

[51] A. Graves, *Supervised Sequence Labelling With Recurrent Neural Networks*, vol. 385. Heidelberg, Germany: Springer, 2012.

[52] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *Proc. AAAI*, 2016, pp. 1230–1236.

[53] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proc. AAAI*, 2016, pp. 2415–2421.

[54] H.-F. Yang, K. Lin, and C.-S. Chen. (2015). "Supervised learning of semantics-preserving hash via deep convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1507.00101

[55] A. Hyvärinen, J. Hurri, and P. O. Hoyer, *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*, vol. 39. Berlin, Germany: Springer, 2009.

[56] T. Tieleman and G. E. Hinton, "Neural networks for machine learning," Coursera, Mountain View, CA, USA, Tech. Rep., 2012.

[57] Y. Lecun and C. Cortes. (1998). *The MNIST Database of Handwritten Digits*. [online] Available: http://vann.lecun.com/exdb/mnist/

[58] G. Irie, Z. Li, X.-M. Wu, and S.-F. Chang, "Locally linear hashing for extracting non-linear manifolds," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2123–2130.

[59] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[60] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 529–534.

[61] D. Song, W. Liu, R. Ji, D. A. Meyer, and J. R. Smith, "Top rank supervised binary coding for visual search," in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 1922–1930.

[62] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proc. Comput. Vis. Pattern Recognit.*, 2010, pp. 3485–3492.

[63] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. (2014). "Return of the devil in the details: Delving deep into convolutional nets." [Online]. Available: https://arxiv.org/abs/1405.3531

[64] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[65] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. ACM Int. Conf. Multimedia*, 2015, pp. 689–692.

[66] Q. V. Le, N. Jaitly, and G. E. Hinton. (2015). "A simple way to initialize recurrent networks of rectified linear units." [Online]. Available: https://arxiv.org/abs/1504.00941

[67] A. Graves. (2013). "Generating sequences with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1308.0850

[68] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. (2017). "Empirical evaluation of gated recurrent neural networks on sequence modeling." [Online]. Available: https://arxiv.org/abs/1412.3555

[69] K. Greff, R. K. Srivastava, J. Koutnìk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[70] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 353–360.

[71] W. Kong and W.-J. Li, "Isotropic hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1655–1663.

[72] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. Dick, "Learning hash functions using column generation," in *Proc. ICML*, 2013, pp. 142–150.

Xiaoqiang Lu (M'14–SM'15) is currently a Full Professor with the Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, China, and also with the University of Chinese Academy of Sciences, Beijing, China. His current research interests include pattern recognition, machine learning, hyperspectral image analysis, cellular automata, and medical imaging.

Yaxiong Chen is currently pursuing the Ph.D. degree with the Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, China, and also with the University of Chinese Academy of Sciences, Beijing, China. His main research interests are pattern recognition, machine learning, and computer vision.

Xuelong Li (M'02–SM'07–F'12) is currently a Full Professor with the Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, China, and also with the University of Chinese Academy of Sciences, Beijing, China.