Deep Progressive Hashing for Image Retrieval

Jiale Bai[1], Bingbing Ni[1], Minsi Wang[1], Yang Shen[1], Hanjiang Lai[2], Chongyang Zhang[1], Lin Mei[3], Chuanping Hu[3], Chen Yao[3]
[1]Shanghai Jiao Tong University, [2]Sun Yat-Sen University,
[3]the Third Research Institute of the Ministry of Public Security

ABSTRACT

This paper proposes a novel recursive hashing scheme, in contrast to conventional one-off based hashing algorithms. Inspired by human's nonsalient-to-salient perception path, the proposed hashing scheme generates a series of binary codes based on progressively expanded salient regions. Built on a recurrent deep network, i.e., LSTM structure, the binary codes generated from later output nodes naturally inherit information aggregated from previously codes while explore novel information from the extended salient region, and therefore it possesses good scalability property. The proposed deep hashing network is trained via minimizing a triplet ranking loss, which is end-to-end trainable. Extensive experimental results on several image retrieval benchmarks demonstrate good performance gain over state-of-the-art image retrieval methods and its scalability property.

KEYWORDS

Deep Hashing, Image Retrieval, Recurrent Neural Networks, Saliency

ACM Reference Format:
Jiale Bai[1], Bingbing Ni[1], Minsi Wang[1], Yang Shen[1], Hanjiang Lai[2], Chongyang Zhang[1], Lin Mei[3], Chuanping Hu[3], Chen Yao[3]. 2017. Deep Progressive Hashing for Image Retrieval. In *Proceedings of MM '17, Mountain View, CA, USA, October 23–27, 2017,* 9 pages. https://doi.org/10.1145/3123266.3123280

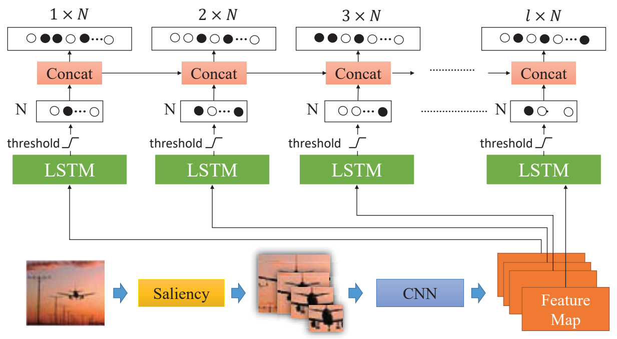


Figure 1: We propose *Deep Progressive Hashing*(DPH), a class of architecture leveraging the complete and comprehensive information extracted from salient regions. The feature extracted by DPH will be more representative since larger sized salient regions enclose smaller ones. The newly generated binary codes inherit discriminative information extracted from previous binary coding steps. The output of DPH is scalable with concatenating the newly binary codes. The number l and N on the top of figure represents the number of concatenated sequence and the number of bits in the sequence respectively.

1 INTRODUCTION

With the growing of image data, more and more research has concentrated on finding more useful methods to make the image retrieval both accurate and efficient. Representing the image with binary codes (known as Hashing [5, 6, 16, 24, 26]), is an efficient method for image search, which maps images with similar semantic information to binary codes with small Hamming distance [27]. Usually, to deal with larger image search database, longer binary codes are utilized, i.e., to make the representation more expressive and discriminative. However, long codes are redundant for small sized problem and consume unnecessary computation and storage resources.

Developing a problem-scalable hashing method is thus highly demanded in multimedia retrieval community [8, 25, 33]. However, most of these methods could be considered as *one-off* solution, namely, the length of the generated binary codes are fixed once and is not feasible to adjust if the problem size changes. In the case of dynamically growth of retrieval database, scalable and adaptive capability is particularly important. To explicitly address this issue, in this paper, we propose Deep Progressive Hashing (DPH), a novel recurrent architecture for scalable image retrieval. This proposed new hashing scheme is capable of recursively generating binary codes from a gradually expanded salient region of the input image, in order to yield finer and finer descriptions to cope with dynamically increasing search database.

Our proposed deep progressive hashing framework is inspired by human's nonsalient-to-salient perception path. More specifically, human visual system can focus on the main object quickly in the image and look around the region of interest. There are two forms of visual attention: bottom-up saliency-driven attention and top-down task-driven attention. Top-down task-driven attention attempts to use one's prior knowledge to process the specific task. Bottom-up saliency-driven attention can grasp the distinctiveness of visual elements and help human to fix their eyes on the main information. Therefore, visual feature extracted from this fixation region could be used to represent the image. Our method is thus inspired by the bottom-up visual perception path. Rather than a fixed region of interest (i.e., salient region), in this work we explore a series of image regions with increasing sizes which all enclose the fixation point (i.e., with larger sized regions enclose smaller ones). More complete and comprehensive discriminative information could be extracted from larger sized salient regions, and since larger sized salient regions enclose smaller ones, feature representations obtained from larger and larger salient regions would become more and more descriptive. This well matches our problem requirements, namely, to deal with small scale retrieval problem, we can just use binary codes computed from high salient regions (i.e., with small region size). And when the problem scales up, we can augment more codes to the existing codes by computing new binary representation from lower salient regions (i.e., with larger region size). The newly computed binary codes which also inherit discriminative information extracted from previous binary coding steps, provide incremental information to describe the image, therefore it could be used to cope with larger sized image search problem.

More specifically, the proposed deep progressive hashing scheme works as follows, as illustrated in Figure 1. First, we leverage off-the-shelf saliency model to generate a series of gradually/progressively expanded salient regions (i.e., with each one encloses its predecessor), therefore forming a sequence of foreground rectangular regions with increasing sizes and decreasing saliency values. Second, each region in the sequence is connected to deep convolutional neural networks based hashing unit, forming basic building blocks of the recursive hashing scheme. Third and most important, to achieve recursive and scalable hashing, we propose a recurrent neural network structure (e.g., LSTM [10]) built on the sequence of salient region based hashing generation units. As an LSTM infrastructure is capable of aggregating discriminative information along the input sequence, binary codes output from later nodes convey useful information extracted with previous output codes, thus yielding more and more discriminative binary encoding. In the same time, it is very flexible to combine arbitrary number of output codes to deal with different sized retrieval problem. The proposed deep hashing network is trained via minimizing a triplet ranking loss, which is end-to-end trainable. Extensive experimental results on several image retrieval benchmarks (NUS-WIDE [14], CIFAR10 [2]) demonstrate our method is state-of-the-art and possesses good scalable property.

The rest of this paper is organized as follows. We review some related work in Section 2. The Deep Progressive Hashing method details are introduced in Section 3. Then we apply DPH in Several benchmarks in Section 4. Finally, conclusion is drawn in Section 5.

2 RELATED WORK

Learning-based Hash. The application of learning-based hash methods in large-scale image retrieval has attracted the interest of researchers because it can perform extremely fast retrieval by mapping the high-dimensional visual data into a low-dimensional binary codes. Learning-based hash approaches can be grouped into three categories: unsupervised, semi-supervised and supervised approaches. Unsupervised approaches use unlabeled data to learn hash functions and preserve similarities in Euclidean space rather than Hamming space. Representative methods are Iterative Quantization (ITQ) [6], Kernelized Locality-Sensitive Hashing (KLSH) [17], and Semantic Hashing(SH) [28]. Semi-supervised approaches [32] generate hash code by labeled and unlabeled information. Supervised approaches [21, 23, 26, 31] learn more significant binary codes by using the labeled data. Binary Reconstruction Embedding (BRE) [15] minimizes the squared error between the Euclidean distance and Hamming distance to learn more efficient binary codes. Minimal Loss Hashing(MLH) [26] minimizes the hinge-like loss for code construction. Supervised Hashing with Kernels (KSH) [23] learns binary codes through minimizing the Hamming distance between the datasets with similar labels while maximizing the Hamming distance between the datasets with dissimilar labels. Other types of work are based on deep neural network [29, 34, 38]. Deep networks can also be used in Deep Hashing (DH) and Supervised DH (SDH) [22]. Deep Multi-View Hashing (DMVH) designs hand-crafted features to handle multi-view data. Deep convolutional networks can also be used in hashing problems. CNNH and CNNH+ [34] design a two-stage learning strategy. They decompose a matrix into hash codes and then use CNN to learn the corresponding hash function. Another method Deep Semantic Ranking Based Hashing (DSRH) [38] optimizes a triplet loss based on labels during hash code construction.

Recurrent Neural Network and LSTM. Recurrent neural networks especially the Long-Short Term Memory (LSTM) [10] have achieved great success in a large variety of applications including temporal modeling such as speech recognition [7], natural language processing, image caption generation [13, 30] and activity recognition [4]. In [4], Donahue *et al* proposed the long recurrent convolutional network (LRCN) which uses the CNN to extract high-level features and stack LSTM on temporal dimension to deal with video recognition and image description tasks.

Saliency model. In past decades, several saliency models [1, 9, 11, 20, 35] have been proposed to simulate the human visual attention. Itti *et al* [11] proposed the bottom-up selective model with fusing color, intensity and orientation of the images. Bruce *et al* [1] use Shannon's self-information to measure the saliency. In [9], Harel *et al* proposed a bottom-up visual saliency model named Graph-Based Visual Saliency. This model consists of two steps: it first builds the activation

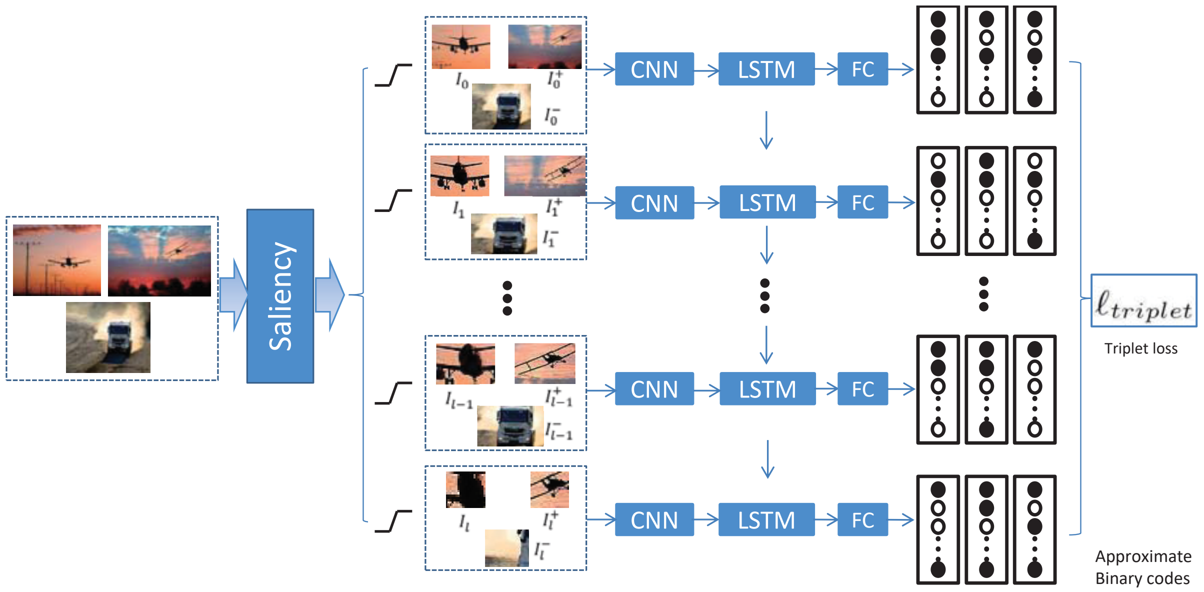


Figure 2: Overview of the Deep Progressive Hashing. DPH randomly selects the images from the training set forming the triplets (I, I^+, I^-). The image I is more similar to the imageI^+ than the image I^-. The saliency model zooms in the main object of the images and generates a series of images with different thresholds. After extracting the semantic features by convolutional neural networks, the next binary codes from later output nodes, with the LSTM, naturally inherit information aggregated from previously codes while explore novel information from the extended salient region. Thus, binary codes contain more information about salient region. The next more accurate codes can be concatenated to the previous codes, which means the DPH is scalable for image retrieval. Finally, the triplet ranking loss preserves the similarities on images.

maps on certain convolutional feature channels, and then normalizes them in a way which highlights conspicuity and admits combination with other maps. Different with these methods, our DPH generates the binary codes by inheriting the previous saliency information and current input, which ensures the new binary codes obtain more efficient information from the salient region.

All of above methods utilize the whole image as input to generate the final binary codes. However, image retrieval is usually to retrieve the part of image (e.g. the object in image). In other words, the key information we want to retrieve usually exists in the part of image rather than the whole image. The information out of interest region(key region) is redundancy for retrieval task. To this end, inspired by human nonsalient-to-salient perception path, we propose a hashing scheme which generates a series of binary codes based on progressively expanded salient regions. Thus, more comprehensive and discriminative binary codes are generated.

3 DEEP PROGRESSIVE HASHING

Conventional Hashing methods attempt to learn a mapping Γ from the original feature space to a binary codes space, i.e., $\Gamma(I)$: $I \to \{0,1\}^N$, such that an input image I can be encoded into N-bit binary codes $\Gamma(I)$. The learned mapping usually preserves semantic similarities, i.e., similar entities in the original space also possess small Hamming distances in the codes space. On the contrary of the above *one-off* solution for binary codes generation, we proposed a recurrent/progressive mechanism for binary codes generation. More concretely, the input is a sequence of image parts/regions (which is obtained from the original image by some way), denoted as $I_1, I_2, ..., I_l$. The output is a sequence of binary codes, denoted as $b_1, b_2...b_l$. (i.e., $\Gamma(I)$: $I \to b$. the sequence $I_1, I_2, ..., I_l$ are encoded into the binary sequence $b_1, b_2...b_l$) l represents the number of images generated from original images. Note that every binary codes in the output sequence is the refined/updated/enhanced version of the previously output codes, with respect to the output order. This progressive scheme possesses the following advantages compared with the traditional one-off binary codes generation. First, DPH progressively generates newly codes inheriting the previous nodes and exploring the novel information from salient region. Second, the recurrent mechanism is also applied to the previous output, in turn, to reduce the redundancy. Finally, DPH possesses the scalability by concatenating the output binary codes.

The proposed progressive hashing scheme is shown in Figure 2. First, the proposed approach gradually zooms in the region of interest based on the corresponding saliency map to generate the sequence inputs. With recursive saliency threshold operation, a series of image patches with increasing amount of details (may with more and more redundancy) are generated. Second, each region in the sequence is connected to deep convolutional neural networks based hashing unit, forming basic building blocks of the problem recursive hashing scheme. Third, an LSTM structure is proposed to take the sequence as the input and recursively generates hashing codes sequence. Note that: in each time step, the output codes inherit information aggregated from previously state while they also explore novel information from the extended salient region, therefore yielding more and more discriminative binary codes. The proposed DPH scheme is scalable and end-to-end trainable.

3.1 LSTM-Hashing Infrastructure

Deep Hashing Unit. The basic hashing unit we adopt in this work is based on [34]. We use the fully connected layer to generate approximate binary codes. Then the output is quantified by a fixed value to generate binary codes. Our basic hashing unit takes the AlexNet for image feature extraction. Lai et al. [18] design deep convolutional neural networks tailored for learning-based hashing. They add max pooling after the convolutional layers with 1×1 kernel and replace the fully-connected layer with average pooling. DNNH [18] designs divide-and-encode module to generate the binary codes. They use a sigmoid activation function followed by a piece-wise threshold function. Then [18] generates the binary codes with threshold. In this paper, we use fully connected layer rather than the divide-and-encode model because the output of LSTM is the semantic information of images. The divided-and-encode module will break the inner relationship of feature. In [18] the average pooling layer uses the 6×6 kernel size to reduce the size of feature map from 6×6 to 1×1, i.e., the final outputs of pooling layer are 1×1 feature maps. In other words, the final feature vector contains no spatial information and can be seen as the representations of different views. The divide-and-encode module will reduce the redundancy while it is not suitable for our method.

Recursive Hashing. Assume we have already obtained a sequence of attended foreground regions from the input image, the purpose of recursive hashing is to aggregate important information from various foreground details thus to update the binary codes step by step to achieve more and more discriminative codes. During recursion, good features$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l\}$ of earlier output are also preserved, yielding a scalable binary coding scheme. More specifically, our recursive hashing scheme is built on the basic hashing unit within the infrastructure of LSTM as described in [37]. Namely, a sequence of basic (deep) hashing units are sequentially linked by LSTM nodes, and the status of each coding unit is propagated to further coding units, along with novel patch information inputs. An LSTM unit consists of the hidden state $h_t \in \mathbf{R}^n$ and a memory cell $c_t \in \mathbf{R}^n$. n is the number of hidden units. LSTMs define that $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid non-linearity and $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1$ is the hyperbolic tangent non-linearity. LSTM unit updates its memory cell and hidden state though 4 gates: input gate $i_t \in \mathbf{R}^n$, forget gate $f_t \in \mathbf{R}^n$, output gate $o_t \in \mathbf{R}^n$, and input modulation gate $g_t \in \mathbf{R}^n$. The LSTM updates for time step t when the inputs are x_t, h_{t-1}, and c_{t-1} are:

$$\begin{aligned} i_t &= \sigma(W_{xi}\mathbf{x}_t + W_{hi}h_{t-1} + b_i) && (1)\\ f_t &= \sigma(W_{xf}\mathbf{x}_t + W_{hf}h_{t-1} + b_f) && (2)\\ o_t &= \sigma(W_{xo}\mathbf{x}_t + W_{ho}h_{t-1} + b_o) && (3)\\ g_t &= \sigma(W_{xc}\mathbf{x}_t + W_{hc}h_{t-1} + b_c) && (4)\\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t && (5)\\ h_t &= o_t \odot tanh(c_t) && (6) \end{aligned}$$

The symbol $\odot$ represents element-wise multiplication. W and b are weights and biases respectively. From Equation 5 , the forget gate f_t decides whether the previous memory cell c_{t-1} is delivered. The input modulation gate g_t is a function of the current input and previous hidden state. The input gate i_t controls how much information of g_t are admitted to c_t. The memory cell c_t modulated by f_t and i_t learns to selectively forget previous information or focus on its current input and delivers them to the next time step. Finally, the output gate o_t determines the number of memory cells transferred to the hidden state. This architecture can incorporate long-term temporal dynamics and complex contextual dependencies.

We simplify the above system with an equation:

$$(h_t, c_t) = LSTM(x_t, h_{t-1}, c_{t-1}) \quad (7)$$

with the help of LSTMs, the newly generated approximate codes e_t $(t = 1, 2, ..., l)$ can selectively forget previous information (h_{t-1}) and receive more object information from the salient region (x_t). Thus, the binary codes are more efficient containing more information about object. In the meantime, as shown in Figure 3, the saliency model moves away from the region of interest. DPH can remember the previous information and low down the weight of the current input, which prevents the previous codes form the influence of error input.

After getting the output of LSTM, given a 256-dimensional feature h_t, the output of fully-connected layer is defined by

$$fc(h_t) = Wh_t \quad (8)$$

where t denotes the timestep of LSTM with W being the weight matrix. Given $m_t = fc(h_t)$, the sigmoid fuction is defined by

$$e_t = sigmoid(m_t) = \frac{1}{1 + e^{-m_t \sigma}} \quad (9)$$

where σ is a hyper-parameter. For one sequence, the outputs of sigmoid are defined as $\{e_1, e_2, ..., e_t\}$ and are quantified by an fixed value, where $t = 1, 2, ..., l$. We use 0.5 as the threshold:

$$b_t = sgn\{e_t, 0.5\} \quad (10)$$

To generate an ordered sequence of image regions to feed into the recursive hashing framework, we utilize visual saliency encoding. The motivation is as follows. Human visual system can sample in detail the most significant area in the image, while it ignores background or uninterested areas. Bottom up saliency-driven attention can help humans to rapidly concentrate on the main object of visual scenes. As shown in Figure 2, we calculate the saliency map via an off-the-shelf graph-based visual saliency model (GBVS) [9]. We then generate a series of images regions $\{I_1, I_2, ..., I_l\}$ by picking out the obtained saliency maps based on increasing threshold values from the original images. Note that the larger the threshold is, the more remarkable the objects are. The sequence generated can be showed as followed:

$$R, C = find\ (saliency\ (I_0) > threshold) \quad (11)$$

$$I = I_0[R_{min} : R_{max}, C_{min} : C_{max}] \quad (12)$$

where I_0 represents the original images, *saliency* denotes the saliency model and *threshold* is the threshold on the output of saliency model. R, C denote the x, y coordinate vector of points in the saliency region respectively. In other words, I is the minimal rectangle region cropped from original image I_0, which contains the salient region.

This series of image regions progressively highlight the object-of-interest: more concentrated foreground regions reflect more object centered information, while larger regions contain more contextual information, and these two types of visual information are complementary to each other. With the proposed LSTM based recursive hashing architecture, important information including foreground object and contextual knowledge are mutually excited to yield more and more discriminative binary codes via information aggregation and noise attenuation.

3.2 Learning DPH

The objective of training the proposed deep progressive hashing network is two-fold. On one hand, following conventional deep hashing optimization goal, similar samples in the original image space should preserve similarity in the encoded binary space. On the other hand, the output binary codes through recursion should possess more and more confidence (i.e., more and more reliable codes through recursion).

Similarity Preserving Objective: Among the current learning-based hashing methods, supervised hashing preserve the pair-wised similarities or triple-wised rankings, devised by the supervised information, to let the binary codes obtain the semantic structure of data more efficiently. The works in [19, 27] preserve relative similarities by forcing the image I is more similar to the image I^+ than the image I^-. The triplet based similarities are easier to capture than the pair-wised similarities.

Our method follows the settings of [19] using triplet loss to preserve relative similarities of images. The input images are formed to the triplet set $(\Gamma(I), \Gamma(I^+), \Gamma(I^-))$. $\Gamma(I^+)$ and $\Gamma(I^-)$ are positive and negative instance respectively. Triplet loss attempts to make the binary code $\Gamma(I)$ closer to $\Gamma(I^+)$ than $\Gamma(I^-)$ in Hamming distance. Let's define the symbol $\|.\|_H$ as the Hamming distance. The triplet loss can be defined as:

$$\begin{aligned}&\ell_{triplet}(\Gamma(I), \Gamma(I^+), \Gamma(I^-))\\ &= \max(0, 1 - (\|\Gamma(I) - \Gamma(I^+)\|_H - \|\Gamma(I) - \Gamma(I^-)\|_H))\end{aligned} \quad (13)$$

The binary constraint on the output leads to a NP-hard optimization problem. Our method relaxes the constraint with ℓ_2 norm and eases the integer with the continuous constraints on $\Gamma(.)$. The new loss can be defined as:

$$\begin{aligned}&\ell_{triplet}(\Gamma(I), \Gamma(I^+), \Gamma(I^-))\\ &= \max(0, 1 + (\|\Gamma(I) - \Gamma(I^+)\|_2^2 - \|\Gamma(I) - \Gamma(I^-)\|_2^2)) \quad (14)\\ &\text{subject to } \Gamma(I), \Gamma(I^+), \Gamma(I^-) \in [0, 1]^N\end{aligned}$$

This relaxation simplifies the optimization problem with a convex loss. We use b, b^+, b^- to represent the $\Gamma(I)$, $\Gamma(I^+)$, $\Gamma(I^-)$. We define the distance $D = 1+(\|b-b^+\|_2^2 - \|b-b^-\|_2^2)$. The indicator function $\delta_{D>0} = 1$ if $D > 0$ while if $D <= 0$, then $\delta_{D>0} = 0$. Thus, for each time step t, the gradients can be solved as:

$$\frac{\partial \ell}{\partial b_t} = (2b_t^- - 2b_t^+) \times \delta_{D_t>0} \quad (15)$$

$$\frac{\partial \ell}{\partial b_t^+} = (2b_t^+ - 2b_t) \times \delta_{D_t>0} \quad (16)$$

$$\frac{\partial \ell}{\partial b_t^-} = (2b_t^- - 2b_t) \times \delta_{D_t>0} \quad (17)$$

Confidence Increasing Objective: As the network's attention progressively narrows down to the object, the measurement for similarity should be more and more confident. In other words, the confidence of LSTM outputs should be monotonically non-decreasing as the newly codes exploring the salient region. However, the triplet ranking loss does not enforce such a monotonous non-decreasing property. We therefore propose a novel monotonous loss, which extends the triplet ranking loss to enforce the accuracy of prediction increase when the time step goes deeper. Mathematically, we can formulate this loss as:

$$L_t^m = \max(0, (-1)^y(s_t - s_t^{pre})), t \neq 0, \quad (18)$$

$$s_t^{pre} = \begin{cases}\max(s_1, s_2..., s_{t-1}), & y = 1,\\ \min(s_1, s_2..., s_{t-1}), & y = 0.\end{cases} \quad (19)$$

where L_t^m denotes the monotonous loss at time-step t, which penalizes the corresponding node if the output similarity score violates the monotonous rule. y is the ground truth label, i.e., 1 for matched and 0 for un-matched. s_t is the predicted similarity score at time step t and s_t^{pre} is the maximum ($y = 1$) or minimum ($y = 0$) prediction score until time step $t - 1$. The max operation of Equation 18 picks out the nodes that violate the monotonous rule. The overall loss could be expressed as:

$$L_t = \ell_{triplet}(\Gamma(I_t), \Gamma(I_t^+), \Gamma(I_t^-)) + \lambda L_t^m, n \neq 0, \quad (20)$$

where λ is a weighting factor for both types of losses.

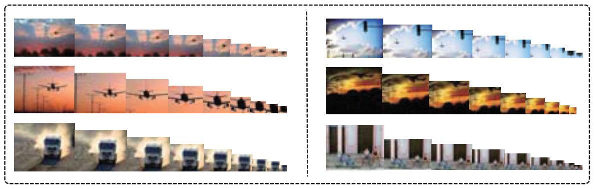


Figure 3: Image sequence generation. Left: the saliency model captures the object well. Right: the saliency model moves away from the region of interest.

3.3 Implementation Details

Data Preparation. DPH uses the saliency model to generate a sequence of images from original image with the different model. We use the GBVS model to generate the salient images and pick out the salient region with different thresholds. As shown in Figure 3, GBVS can find the object of the images when it moves to the area which is highly contrast with the background sometimes. We generate 5-images sequence for CIFAR-10 [14] and 9-images sequence for NUS-WIDE [2].

Network Parameter. DPH is based on the open source Caffe [12] framework. This network is trained by stochastic gradient descent with 0.9 momentum. The weight decay parameter is 0.0005 and gamma is 0.1. The batch is composed by the timestep t and the number of sequence N. The timestep represents the number of images generated from one image. In this paper, we set batch as (9,32) for NUS-WIDE and (5,60) for CIFAR-10. We fine tune our network with the fixed weights based on the AlexNet model pretrained on ILSVRC 2012. After the loss is steady, learning rate is set to 0.01 at the beginning and decreased once every 10000 iterations.

4 EXPERIMENTS

4.1 Experiments setting

We test our method on several benchmark datasets:

- NUS-WIDE: The NUS-WIDE dataset provides about 270000 URLs of images collected from Flickr. The images in NUS-WIDE have more than one label taken from 81 concept tags. We collect 269646 images from the Internet. We follow the settings in [18] to use the images with 21 most frequently used labels. There are at least 5000 images for each label.
- CIFAR-10: The CIFAR dataset is composed of 60000 32×32 color images grouped into 10 classes. Each class has 6000 images.

In NUS-WIDE, for each class, DNNH [18] randomly selects 100 images to form the test set and 500 images excluding the test set as training set. Totally, the test set contains 2100 images and the training set contains 10500 images. We use the rest images and the training set as the retrieval data. SSDH [36] obtains about 230000 images from the given links and divides the data set into two parts: a training set of 97214 images and a test set of 65075 images. SSDH [36] uses 97214 images as the training set and selects 2100 images from the test set to form the query set. We follow the DNNH [18], BOH-CNN [3] and compare with other methods. BOH-CNN [3] uses the VGGNet to extract the semantic feature, while DPH uses the AlexNet which has the same level extracting feature as DNNH [18]. In CIFAR-10, we randomly select 1000 images, 100 images per class, from the test set to form the test query. The training set contains 5000 images selected from the rest images.

We compare the performance with other state-of-the-art hashing methods including supervised methods (e.g. DNNH, CNNH, KSH and ITQ-CCA) and unsupervised methods (e.g. MLH, BRE, ITQ, SH and LSH). Our method is the state-of-the-art in different evaluation metrics with the same feature extraction structure (i.e. CNN with similar ability). BOH-CNN [3] extracts feature by VGGNet while DPH extracts the feature with AlexNet. SSDH [36] uses 97214 images for training in NUS-WIDE and 50000 images for training in CIFAR-10 while DPH uses 10500 images for training in NUS-WIDE and 5000 images for training in CIFAR-10.

We use the metrics as described in CNNH [18] and D-SRH [38]. We use Mean Average Precision(MAP), precision - recall curves, precision curves within Hamming distance 2, precision curves w.r.t. different numbers of top returned samples, Weighted MAP and ACG(100). For fair comparison, we use the identical training and test sets for all of the methods.

4.2 Results of Search Accuracies

Table 1 and Figure 4, 5 show the performance of all the methods in CIFAR10 and NUS-WIDE. Our method DPH-CONCAT achieves the best performance. DPH-FIRST uses the first binary output generated by the original image for test. It is still superior over the others in all criterions with the same testing time. For example, DPH-FIRST obtains relative increase of 19% $\sim$ 22% on CIFAR-10(MAP) and 7.6% $\sim$ 6.4% on NUS-WIDE(MAP). DPH-CONCAT gets the increase of 24.8% $\sim$ 25.4% on CIFAR-10(MAP) and 8.6% $\sim$ 9.5% on NUS-WIDE(MAP). Figure 4 shows that our method achieves great improvement compared with the DNNH in all evaluation metrics. These results demonstrate that our method can grasp the object of images in CIFAR-10 and generate efficient binary codes achieving excellent retrieval performance. As shown in Figure 3, saliency model picks out the region enclosing wrong object or moves to another object as the threshold becomes larger. Thus, LSTMs in our method prevent the previous binary codes from the influence of salient region enclosing wrong object. Figure 5 shows that our method is better than DNNH on MAP and top-k precision while it has comparable performance as the DNNH on precision-recall. These results demonstrate that our method obtains improvement on NUS-WIDE although the sequence contains some images without object. We use evaluation metrics ACG(100) and weighted MAP described in [38]. DSRH [38] collects 226265 images and randomly selects 5000 images for testing queries and the rest is used

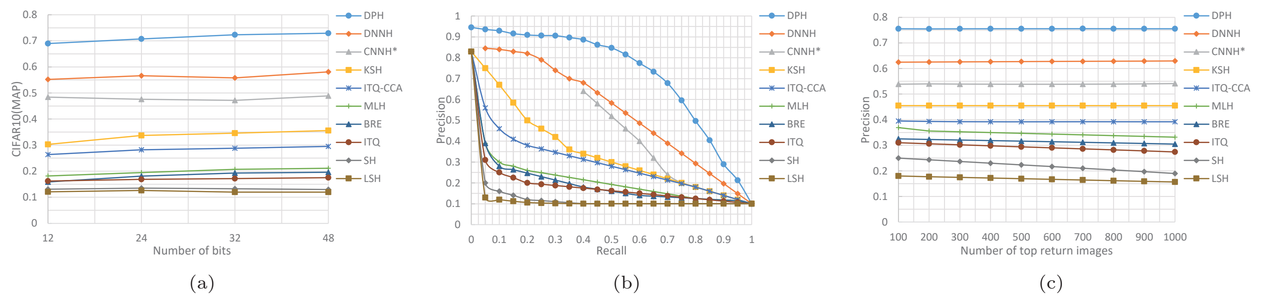


Figure 4: The comparison results on CIFAR10. (a) precision curves within Hamming radius 2; (b) precision-recall curves of Hamming ranking with 48 bits; (c) precision curves with 48 bits w.r.t different numbers of top returned samples.

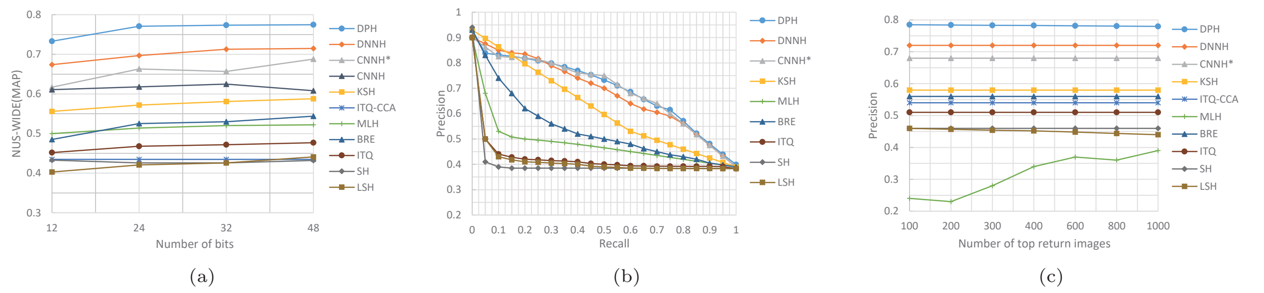
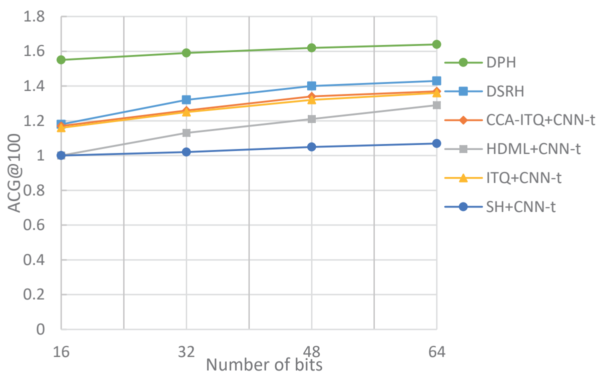


Figure 5: The comparison results on NUS-WIDE. (a) precision curves within Hamming radius 2; (b) precision-recall curves of Hamming ranking with 48 bits; (c) precision curves with 48 bits w.r.t different numbers of top returned samples

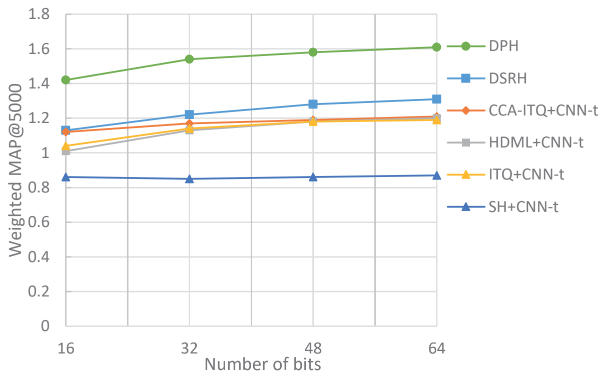
Table 1: MAP of Hamming ranking w.r.t different numbers of bits on CIFAR-10 and NUS-WIDE. For NUS-WIDE, we calculate the MAP values within the top 5000 returned neighbors. The results are directly cited from the respective paper. DPH-FIRST represents the binary codes are generated from the original images without concatenation. DPH-CONCAT represents all the binary codes generated by concatenating the output.

Method	CIFAR-10(MAP)				NUS-WIDE(MAP)			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
DPH-CONCAT	**0.689**	**0.707**	**0.716**	**0.729**	**0.734**	**0.771**	**0.775**	**0.783**
DPH-FIRST	0.662	0.687	0.693	0.709	0.725	0.749	0.751	0.761
DNNH [18]	0.552	0.566	0.558	0.581	0.674	0.697	0.713	0.715
CNNH* [18]	0.484	0.476	0.472	0.489	0.617	0.663	0.657	0.688
CNNH [34]	0.439	0.511	0.509	0.522	0.611	0.618	0.625	0.608
KSH [23]	0.303	0.337	0.346	0.356	0.556	0.572	0.581	0.558
ITQ-CCA [6]	0.264	0.282	0.288	0.295	0.435	0.435	0.435	0.435
MLH [26]	0.182	0.195	0.207	0.211	0.500	0.514	0.520	0.522
BRE [15]	0.159	0.181	0.193	0.196	0.485	0.525	0.530	0.544
SH [28]	0.131	0.135	0.133	0.130	0.433	0.426	0.426	0.423
ITQ [6]	0.162	0.169	0.172	0.175	0.452	0.468	0.472	0.477
LSH [17]	0.121	0.126	0.120	0.120	0.403	0.421	0.426	0.441

for training and retrieval. As shown in Figure 6, we use larger dataset containing 269646 images and use 10500 images for training, which means we get excellent performance on ACG(100) and weighted MAP with the top 5000 returns in a larger search space and smaller training set. These results verify the DPH not only learns significant representation but also generates efficient binary codes leading to more satisfactory ranking performance.



(a)



(b)

Figure 6: Comparison of ranking performance of our DPH and other hashing methods based on activation features of fine-tuned CNN on NUS-WIDE.

Table 2: MAP of Hamming ranking w.r.t different concatenated numbers of bits on CIFAR-10 and NUS-WIDE. For NUS-WIDE, we calculate the MAP values within the top 5000 returned neighbors.

Number	CIFAR-10(MAP)				NUS-WIDE(MAP)			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
9	N/A				**0.734**	**0.771**	**0.775**	**0.783**
8	N/A				0.734	0.770	0.774	0.771
7	N/A				0.733	0.768	0.772	0.770
6	N/A				0.731	0.767	0.770	0.769
5	**0.689**	**0.707**	**0.716**	**0.729**	0.730	0.764	0.768	0.767
4	0.686	0.695	0.709	0.725	0.728	0.763	0.766	0.765
3	0.685	0.695	0.701	0.720	0.727	0.762	0.763	0.763
2	0.681	0.694	0.698	0.723	0.727	0.760	0.760	0.761
1	0.662	0.687	0.693	0.709	0.725	0.749	0.751	0.753

4.3 Results of Scalable Binary Codes

We test the different length of the binary codes concatenating the progressive output. As shown in Table 2, the first column represents how many outputs are concatenated. The bigger the number is, the higher the MAP in both CIFAR-10 and NUS-WIDE are. These results indicate that our method is variable for different request. On one hand, the performance of DPH-FIRST is superior over others with the same time consumption. On the other hand, if the problem is not so urgent to the requirement of time, our scalable output can satisfy the different requests to improve the performance costing more testing time.

5 CONCLUSION

In this paper, we develop deep progressive hashing method for image retrieval, which generates a series of binary codes based on progressively expand salient regions. Based on the recurrent neural network, the newly generated codes inherit information aggregated from previous codes and explore the salient region. Thus, the output of DPH possesses excellent scalability property satisfying different problems. The proposed network is an end-to-end training via minimizing a triplet ranking loss. The output of DPH is quantified by a fixed value to generate binary codes. The different criterions in image retrieval show that our method deep progressive hashing method is state-of-the-art and is scalable output satisfying different application scene.

ACKNOWLEDGMENTS

This work was supported by NSFC (U1611461, 61502301, 61527804), China's Thousand Youth Talents Plan, and the Shanghai Key Laboratory of Digital Media Processing and Transmissions. This work was partially supported by the Opening Project of Shanghai Key Laboratory of Digital Media Processing and Transmission.

REFERENCES

[1] Neil Bruce and John Tsotsos. 2006. Saliency based on information maximization. *Advances in neural information processing systems* 18 (2006), 155.
[2] Tat Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. NUS-WIDE: a real-world web

image database from National University of Singapore. In *ACM International Conference on Image and Video Retrieval*. 48.

[3] Qi Dai, Jianguo Li, Jingdong Wang, and Yu-Gang Jiang. 2016. Binary Optimized Hashing. In *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*. 1247–1256. https://doi.org/10.1145/2964284.2964331

[4] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2625–2634. https://doi.org/10.1109/CVPR.2015.7298878

[5] Yunchao Gong, Sanjiv Kumar, Henry A. Rowley, and Svetlana Lazebnik. 2013. Learning Binary Codes for High-Dimensional Data Using Bilinear Projections. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*. 484–491.

[6] 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 Trans. Pattern Anal. Mach. Intell.* 35, 12 (2013), 2916–2929.

[7] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*. 6645–6649.

[8] Yun Gu, Chao Ma, and Jie Yang. 2016. Supervised Recurrent Hashing for Large Scale Video Retrieval. In *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*. 272–276. https://doi.org/10.1145/2964284.2967225

[9] Jonathan Harel, Christof Koch, Pietro Perona, et al. 2006. Graph-based visual saliency. In *NIPS*, Vol. 1. 5.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[11] Laurent Itti, Christof Koch, and Ernst Niebur. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence* 20, 11 (1998), 1254–1259.

[12] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*. 675–678. https://doi.org/10.1145/2647868.2654889

[13] Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 3128–3137.

[14] Alex Krizhevsky. 2012. Learning Multiple Layers of Features from Tiny Images. (2012).

[15] Brian Kulis and Trevor Darrell. 2009. Learning to Hash with Binary Reconstructive Embeddings. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.* 1042–1050.

[16] Brian Kulis and Kristen Grauman. 2009. Kernelized locality-sensitive hashing for scalable image search. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*. 2130–2137.

[17] Brian Kulis and Kristen Grauman. 2012. Kernelized Locality-Sensitive Hashing. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 6 (2012), 1092–1104.

[18] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 3270–3278.

[19] Xi Li, Guosheng Lin, Chunhua Shen, Anton van den Hengel, and Anthony R. Dick. 2013. Learning Hash Functions Using Column Generation. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. 142–150.

[20] Yin Li, Xiaodi Hou, Christof Koch, James M Rehg, and Alan L Yuille. 2014. The secrets of salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 280–287.

[21] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and David Suter. 2014. Fast Supervised Hashing with Decision Trees for High-Dimensional Data. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. 1971–1978.

[22] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep hashing for compact binary codes learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2475–2483.

[23] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*. 2074–2081.

[24] Xianglong Liu, Junfeng He, Bo Lang, and Shih-Fu Chang. 2013. Hash Bit Selection: A Unified Solution for Selection Problems in Hashing. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*. 1570–1577.

[25] Viet Anh Nguyen, Jiwen Lu, and Minh N. Do. 2014. Supervised Discriminative Hashing for Compact Binary Codes. In *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*. 989–992. https://doi.org/10.1145/2647868.2655003

[26] Mohammad Norouzi and David J. Fleet. 2011. Minimal Loss Hashing for Compact Binary Codes. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. 353–360.

[27] Mohammad Norouzi, David J. Fleet, and Ruslan Salakhutdinov. 2012. Hamming Distance Metric Learning. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.* 1070–1078.

[28] Ruslan Salakhutdinov and Geoffrey E Hinton. 2007. Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure.. In *AISTATS*, Vol. 11.

[29] Ruslan Salakhutdinov and Geoffrey E. Hinton. 2009. Semantic hashing. *Int. J. Approx. Reasoning* 50, 7 (2009), 969–978.

[30] O Vinyals, A Toshev, S Bengio, and D Erhan. 2015. Show and tell: A neural image caption generator. *Computer Science* (2015), 3156–3164.

[31] Jun Wang, Wei Liu, Andy X. Sun, and Yu-Gang Jiang. 2013. Learning Hash Codes with Listwise Supervision. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*. 3032–3039.

[32] Qifan Wang, Luo Si, and Dan Zhang. 2014. Learning to Hash with Partial Tags: Exploring Correlation between Tags and Hashing Bits for Large Scale Image Retrieval. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*. 378–392.

[33] Botong Wu and Yizhou Wang. 2016. Neighborhood-Preserving Hashing for Large-Scale Cross-Modal Search. In *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*. 352–356. https://doi.org/10.1145/2964284.2967241

[34] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. 2014. Supervised Hashing for Image Retrieval via Image Representation Learning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.* 2156–2162.

[35] Junchi Yan, Mengyuan Zhu, Huanxi Liu, and Yuncai Liu. 2010. Visual saliency detection via sparsity pursuit. *IEEE Signal Processing Letters* 17, 8 (2010), 739–742.

[36] Huei-Fang Yang, Kevin Lin, and Chu-Song Chen. 2015. Supervised Learning of Semantics-Preserving Hashing via Deep Neural Networks for Large-Scale Image Search. *CoRR* abs/1507.00101 (2015).

[37] Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615* (2014).

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