Sakarya University Journal of Computer and Information Sciences
http://saucis.sakarya.edu.tr/

DOI: 10.35377/saucis…1339150 RESEARCH ARTICLE

Sequential and Correlated Image Hash Code Generation with Deep Reinforcement Learning

Can Yüzkollar [1]

[1]Computer Engineering Department, Faculty of Computer and Information Sciences, Sakarya University, Sakarya, Türkiye

Corresponding author:
Can Yüzkollar, Computer Engineering Department, Faculty of Computer and Information Sciences, Sakarya University, Sakarya, Türkiye
E-mail address:
can@sakarya.edu.tr

Submitted: 07 August 2023
Revision Requested: 28 August 2023
Last Revision Received: 29 August 2023
Accepted: 31 August 2023
Published Online: 31 August 2023

Citation: Yuzkollar C. (2023). Sequential and Correlated Image Hash Code Generation with Deep Reinforcement Learning. *Sakarya University Journal of Computer and Information Sciences.* 6 (2) https://doi.org/10.35377/saucis...1339150

ABSTRACT

Image hashing is an algorithm used to represent an image with a unique value. Hashing methods, which are generally developed to search for similar examples of an image, have gained a new dimension with the use of deep network structures and better results have started to be obtained with the methods. The developed deep network models generally consider hash functions independently and do not consider the correlation between them. In addition, most of the existing data-dependent hashing methods use pairwise/triplet similarity metrics that capture data relationships from a local perspective. In this study, the Central similarity metric, which can achieve better results, is adapted to the deep reinforcement learning method with sequential learning strategy, and successful results are obtained in learning binary hash codes. By considering the errors of previous hash functions in the deep reinforcement learning strategy, a new model is presented that performs interrelated and central similarity-based learning.

Keywords: Image Hashing, Reinforcement Learning, Deep Reinforcement Learning

1. Introduction

Today, millions of images are transferred to the web daily due to the increase in internet speeds and developments in web technologies. Due to this increase in the number of images on the web and the difficulties in accessing targeted images, many studies have been conducted to improve image search processes [1-8].

In visual search processes, a unique code representing an image and hash codes are usually generated using an algorithm and operations are performed on the codes. Since similar codes are generated for similar images, the similarity of the images can be easily detected. In traditional cryptographic algorithms, changes that do not perceptually distort the image (such as changing some pixel values, resizing the image, etc.) will generate different codes due to the nature of these algorithms. Therefore, the use of such algorithms is not suitable for image similarity detection. Some hashing methods use traditional hand-crafted feature extraction and deep hashing methods use deep network structures to generate hash codes. In hashing approaches, converting the input images into compact binary codes provides time and memory utilization advantages in image search processes.

Image hashing algorithms aim to represent images with short hash codes. They are frequently used for detecting similar images or content, checking for copyright violations, performing fast and efficient searches in large media libraries, creating summaries of sensitive image data to hide the original image, providing biometric recognition such as face recognition, etc. These algorithms play a crucial role in areas ranging from content similarity detection to security measures.

Traditional methods using handcrafted features could be more efficient in expressing visual content, which has a negative impact on performance in the visual search process. The high performance of deep learning methods in image classification

and object recognition has led to using these methods in hash code generation. Some deep hashing methods have been developed that utilize the feature extraction power of deep networks. These hashing methods treat all hash functions independently and do not consider the correlations of different hash functions. These studies usually utilize pairwise or triplet data similarity [9-11]. Accordingly, loss calculations are performed. It has been stated in some studies that the sequential handling of hash codes will provide error correction capability in the learning process. Another study emphasized that the central similarity metric gives more successful results than triplet and pairwise metrics [12].

Deep reinforcement learning has achieved human-like performance in various studies and has been addressed in many fields. A standard reinforcement learning model consists of an environment and an agent interacting. [13] The agent evaluates the information coming from the environment and aims to choose an action that maximizes the sum of future rewards. In hash methods, if we consider the function to obtain the hash code as the agent, the actions generated can be expressed as 0 or 1. Rewards can be obtained by using a loss function to evaluate the quality of the generated hash code information. Maximization of these rewards can be achieved through reinforcement learning. Determining the reward function in a reinforcement learning problem is one of the most important problems. In a study conducted in this direction, the reward function was considered a triplet loss, which was found to give successful results [14]. As stated in [12], the central similarity loss function has a more discriminative aspect than triplet loss and pairwise loss. Our study proposes a deep reinforcement learning method that provides the sequential decision-making process of hash functions and the central similarity reward function.

This paper proposes a deep reinforcement learning model with a sequential learning strategy that considers global data distribution, data relationships and global similarity. The proposed model includes a feature extraction network and a policy network. In the policy network, the RNN structure is used to obtain the probabilities of the binary transformation of images into binary codes sequentially. In the proposed Sequential and Correlated Image Hash Code Generation with Deep Reinforcement Learning (SCIHCGRL) method, the central similarity is suggested as a reward function instead of pairwise or triplet functions used in previous studies.

2. Related Works

Currently used hashing methods are classified into two main groups: data-dependent and data-independent. Data-dependent hashing methods can also be categorized into supervised, unsupervised, and reinforcement. Data-independent methods are based on the nearest-neighbor search method. Peng and Indyk presented Locality-Sensitive Hashing (LSH) [14, 15], where the basic idea is to summarize data and query points so that the probability of collision is much higher for points close to each other than for points far apart. The LSH method requires long codes or multiple hash tables. Lv [16] and Raginsky [17] have proposed several approaches to overcome these problems.

Data-dependent methods are generally divided into two main groups: supervised methods where data labels are used in the learning process and unsupervised methods where data labels are not considered in the learning process. Reinforcement learning methods, in which data labels are used only for evaluation purposes and the learning strategy is not supervised, can also be considered a separate class.

Studies on unsupervised methods are as follows: Spectral hashing (SH)[18], iterative quantization (ITQ)[19], topology preserving hashing (TPH)[30], locally linear hashing (LLH)[20], discrete graph hashing (DGH)[21], scalable graph hashing (SGH)[22] ordinal embedding hashing (OEH)[23], semi-paired discrete hashing (SPDH)[24], similarity adaptive discrete deep hashing (SADH)[25], multiview discrete hashing (MDH)[26], isotropic hashing (IsoH)[27]. In recent years, supervised methods have been used instead of unsupervised methods with more successful results.

To strengthen feature extraction and feature learning in supervised methods, deep learning structures have started to be used and have improved classification performance by characterizing the non-linear features of the data more effectively. Based on the convolutional neural network structure CNNH [28], a two-stage method is preferred in this study. In the first stage, approximate hash code learning was performed by preserving pairwise semantic information and in the second stage, deep hash network training was performed using the learned hash codes as labels. The independence of the deep hash network from the first stage limits the derivation of better hash codes. To overcome this problem, a network within a network (NINH) hash function and image features are presented as simultaneous inputs [29]. Furthermore, various studies use Deep Hashing methods [9,30-34].

Yuan et al. presented a new general similarity metric in [12], which offers a different approach to methods that aim to learn hash functions based on pairwise and triplet data relations. It has been stated that the use of pairwise and triplet-based functions uses only partial relationships between data pairs and may damage the identifiability of the generated hash codes, may have low efficiency in unbalanced data, low efficiency in generating profile similarity among the entire data set and high temporal complexity. It was observed that the proposed central similarity method achieved more successful results.

In this study, a deep reinforcement hashing method using the Central similarity metric is developed.

2. Proposed Method

Most image-hashing studies in the literature have been performed with supervised learning methods. In this paper, we implement a new methodology for generating image hashes based on the methodology described in [14]. This work generates image hashes with several deep networks and deep reinforcement learning.

In our proposed deep reinforcement learning-based approach, RNN network structures are chosen as an agent modeling the hash functions so that the previous hash functions' errors are considered during sequentially transferring images to binary codes. In addition, the Central Similarity metric, which can achieve more successful results, was adapted to the model and more effective results were obtained in learning binary hash codes.

Reinforcement learning involves a challenge encountered by an agent that needs to acquire behavior by engaging in trial-and-error interactions within a dynamic environment [13,35]. Within the conventional framework of deep reinforcement learning, the agent takes the current environmental state as input and produces an output action. This action influences the state of the environment, and the environment communicates with the agent using a scalar reinforcement signal referred to as a reward, which signifies the actions' effectiveness. The reinforcement learning objective is to train the agent to select actions that maximize the cumulative reward. Significant advancements have been made in various domains through deep reinforcement learning techniques.

The deep reinforcement learning we used in this study is shown in Figure 1. In this network, the agent performs actions with stochastic policies and tries to maximize the amount of future reward with a reward-punishment system for each action. In reinforcement deep learning, the policy, state, action, and reward must be determined. In this study, the feature layer output of a CNN network and the historical information of an RNN network are used for the state. For the policy and actions to be implemented, an RNN network and a linear deep network layer are nested. An internal reward system is prepared for each action taken.

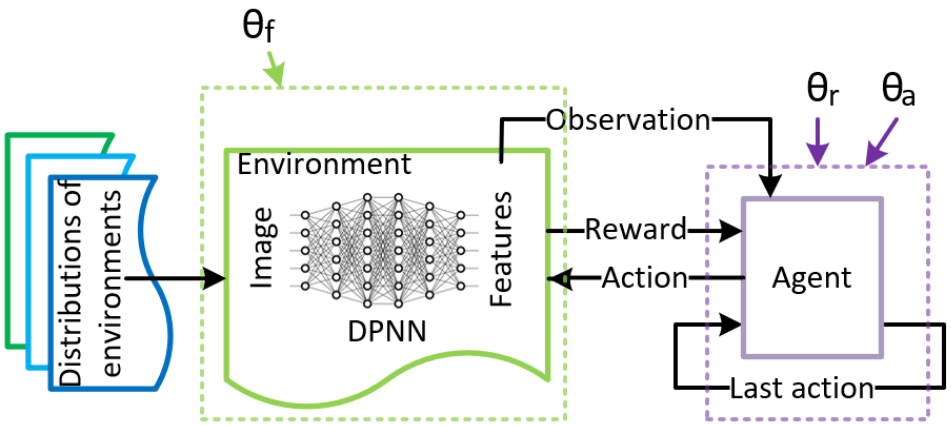

Figure 1 The deep reinforcement learning architecture for image hashing.

States:
A state (s), is a binary given in the output of Figure 2 and is defined as $s = (fv, hv)$. where fv is the image feature vector and hv is the background information of the hash codes. In a pre-trained CNN network, the feature vectors of the original images are extracted. The historical information is obtained from the policy layer.

The first 18 layers of the VGG-19 network were used as is to generate the fv in the state. The last fully connected layer output of the VGG-19 network is used as image features. The structure of this VGG-19 network is shown in Figure 3.

Actions:

There are two possible actions in a hashing problem. These actions are a=1 bit or a=0 bit. Therefore, the probability sum of the possible actions will be 1. This study, -1 is used instead of the 0-action resulting from the policy layer. The agent used generates the probability of action between -1 and 1 as the output of the policy layer.

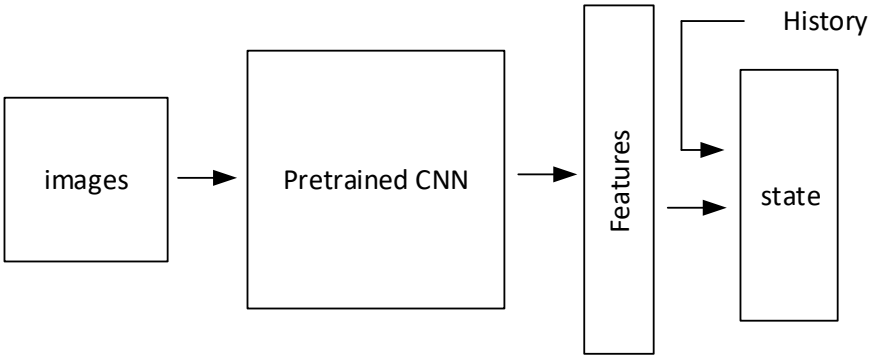

Figure 2 Generation State

$$P(a|s,\theta) = \begin{cases} 1 - policy(s,\theta) & a = -1 \\ policy(s,\theta) & a = 1 \end{cases} \quad (1)$$

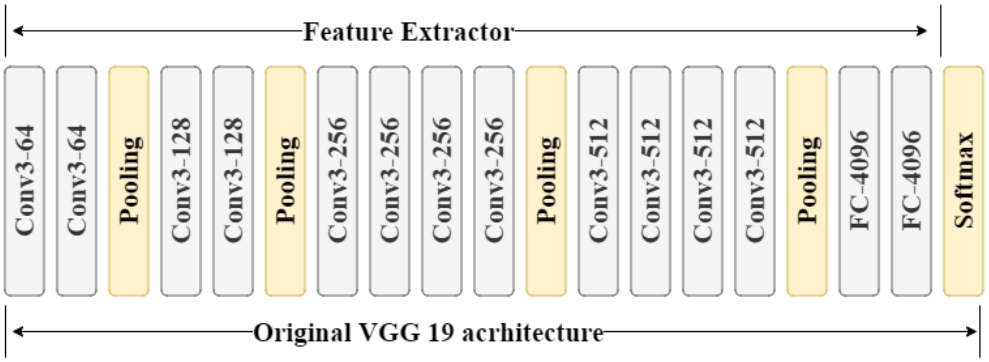

Figure 3 VGG – 19 Network Architecture

The probability distribution of the policy layer is given in Equation 1. $policy(s,\theta)$ denotes the output of the policy layer, s,θ denote the feature vector and the weights of the deep network respectively.

In classical deep learning-based hashing methods, the output layer's output is the number of hash lengths to be obtained. This work generates a sequential action using the past hash information. A policy layer can only output one bit of action. However, a one-bit hash code does not provide enough information to correct previous action errors.

In other words, generating a hash sequence where each bit is generated independently or generating it as a group independently reduces the correlation between the hash bits. Therefore, the previous action errors will be eliminated if the correlation between hash bits is utilized to generate hash bits. The RNN layer in our agent provides this. Instead of a one-bit hash code in our work, we used a sequential action with reduced correlation between pixels.

$$A_t^i = \{a_1, a_2, \dots, a_k\} \quad (2)$$

In short, an action in reinforcement learning is a set of actions in our study.

Rewards:

An action set is rewarded with a single reward. Each action set is a set of k-ordered hash functions. A_t^i is the i-th action set at time t. k is the number of actions and corresponds to the desired hash length. From probability theory, the action probability of a set of k ordered actions are defined as $P(A_t^i|s_t^i,\theta)$.

$$P\left(A_t^i|s_t^i,\theta\right) = \prod_{j=1}^{k} P(a_j|\hat{s}_j,\theta) \quad (3)$$

s_t^i is the i-th state at time t, a_j is an element of the action setA_t^i and $\hat{s}_j$ is the current input state entering the policy layer. The details will be discussed in the next section.

A reward function was created for the performance of each action taken. In creating this function, the concept of hash center in [12] is used. For a dataset with m classes, m hash centers are created. We try to force the hash codes generated for classes in similar groups to resemble the hash centers generated for the class.

The hash center for all classes is defined as $HC = \{c_1, c_2, \dots c_m\}$ and c_i is defined as a k-element array of {-1,1}. Equation 4.1 is a reward function for action set i at time t,

$$J_t^i(A_t^i, c_i) = 1 - BCE(A_t^i, c_i) \tag{4.1}$$

$$BCE = L(\theta) = -[A_t^i * Log(c_i) + (1 - A_t^i) * log(1 - c_i)] \tag{4.2}$$

Equation 4.2, BCE is a creation that calculates the Binary Cross Entropy between the target and input probability. In this study input is model output and target is generated with central similarity hash codes.

The pseudo-code for creating the hash center is given in Algorithm 1.

Algorithm 1 Generation of Hash Center [12]

Input : The number of hash centers m, the dimension of the Hamming space (hash codes) K.
Initialization: construct a $K \times K$ Hadamard matrix $H_K = [h_a^i]$ and construct $H_{2K} = [H_K, -H_K]^\top = [h_{2k}^i]$.
for *iteration* i, i=1 *to* m **do**
  if $m \leqslant K$ & $K = 2^n$ **then** // n is any $\mathbb{Z}^+$
    $c_i = h_a^i$;
  end
  else if $K < m \leqslant 2K$ & $K = 2^n$ **then**
    $c_i = h_{2k}^i$;
  else
    c_i[random half position] $= 1$;
    c_i[other half position] $= 0$;
  end
end
Replace all -1 with 0 in these centers;
Output: hash centers: $\mathcal{C} = \{c_1, \dots, c_m\} \subset \{0,1\}^K$.

For single-label datasets, hash centers are created as in Algorithm 1. For multi-label datasets, hash center $\{c_1, c_2, \dots c_m\}$ is created for each class. If an image is associated with two or more labels (classes), the hash center centroid of the corresponding image is used. For example, a dataset contains $\{l_1, l_2, \dots l_m\}$ labels. If an image has a label such as l_1, l_3, l_6, the centroid of these three centers is calculated by using the hash centers c_1, c_3, c_6 of these labels. In the multi-label dataset, the generated hash center centroid is used instead of c_i in the reward function.

Two sequential rewards are used to find correct hash codes. The first one, R_t^i, is the internal cluster reward for the i-th action set and is concerned with hash code accuracy at the cluster level. The second reward R^i is the global reward and is concerned with the accuracy of all hash codes of image i.

$$\begin{aligned} R_t^i &= -J_t^i(A_t^i, c_i) \\ R^i &= -J^i(A^i, c_i) \end{aligned} \tag{5}$$

A. Deep Reinforcement Learning Approach for Image Hashing

The general structure of the study in Figure 4 shows the Agent and Environment, which are the components involved in reinforcement learning. The Environment prepares image features and hash center representations and executes the reward function. The Agent contains a policy layer and executes the implementation of the policy. The state is formed with the image feature obtained from the Environment and the history information formed in the policy layer.

Image features and hash centers are obtained in Environment with VGG19 and Algorithm 1. A single RNN cell is used in the policy network. RNN cells separate themselves from normal neurons in that they have a state and thus can remember information from the past. The RNN converts image features into local states. By applying a linear layer to the local states, policy values between -1 and 1 are obtained. The basic idea behind the application of RNN is to continue the training without ignoring the past information and to generate the actions in this process.

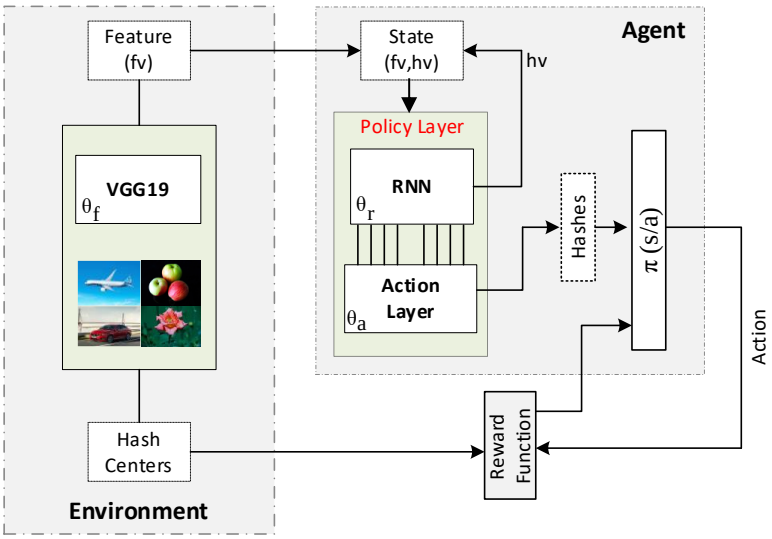

Figure 4 Proposed Method Schema

The schematic of the policy network created for generating actions as a set and reducing the correlation of actions is given in Figure 5. Here, instead of an RNN network, a single RNN cell is used to generate actions with the desired number of hashes. Background knowledge is applied in this part to eliminate previous action errors. The input of the RNN memory cell is the current input image features x_t and the previous RNN cell's h_{t-1} history information obtained from the previous step. The RNN cell generates the desired number of hashes (k) from a single RNN cell using the background information obtained from its inputs and an inner loop. A single action is obtained by applying a linear layer to the h_n obtained in each loop. At the output of the loop, the current RNN cell history h_t and k actions are generated.

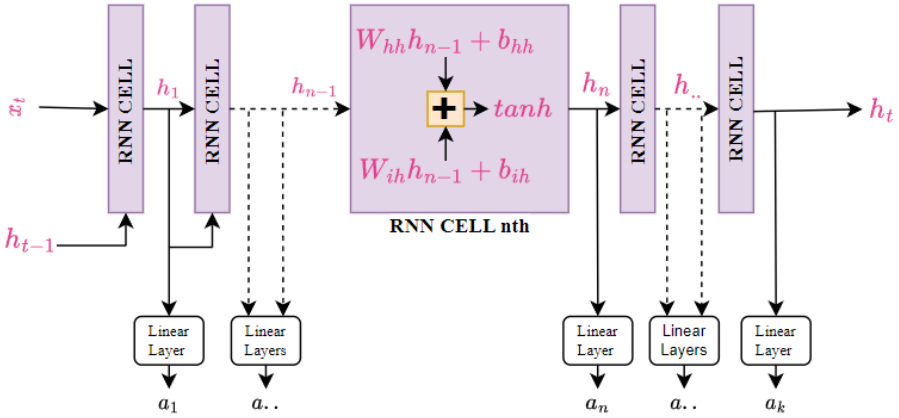

Figure 5 Diagram of the policy network where actions are generated as a cluster.

For (x_t, h_{t-1}) as a state, the RNN layer converts the inputs into outputs with the activation function applied to the cell. Equation 6 is used for the first cycle and Equation 7 for the other cycles.

$$h_n = \tanh\left(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh}\right) \quad (6)$$

$$h_n = \tanh\left(W_{ih}h_{n-1} + b_{ih} + W_{hh}h_{n-1} + b_{hh}\right) \quad (7)$$

The output of the policy network is the action set and the final history is $h_t = h_n$. In the equations x_t and h_t are the input and hidden vectors, where t is the t th step value. W_{ih} is the weight matrix generated for the input values x_t and h_{n-1}. W_{hh} are weight matrices generated against h_n. b_{ih} and b_{hh} are bias terms.

In the given policy network, the input image and the past RNN memory cell are used in the first state (first cycle) to generate actions for an image. In other cases (cycles), the image features information can be stored in the RNN hidden states and

replace past action information. This way, historical information is utilized to generate hash codes and correction of past action errors is realized. The h_t at the last stage of the cycle gives the history information for the next steps in Figure 1.

The policy layer, the linear layer, which is the fully connected layer, is defined as in Equation 8. The output of the linear layer includes a linear network and an activation function.

$$a_n(x) = \tanh\left(W_h^T h_n + b\right) \tag{8}$$

h_n is the RNN layer output, W_h^T is the policy layer weights and v is the bias term. At each step t, the output of the linear policy network is mapped between -1 and 1. With a threshold value, the map values are converted into hash codes. In this study, the threshold value will be 0 since probability values between -1 and 1 will occur. ($i = 1 \dots k$)

$$b_i(x) = \begin{cases} -1 & a_i(x) < 0 \\ 1 & a_i(x) \geq 0 \end{cases} \tag{9}$$

B. Environment vs Agent

In our study, three networks are trained. A stochastic gradient descent algorithm is used to update the parameters of the networks. The first network is used in the CNN network in the environment. The training parameter. θ_f is updated at each step. While in other deep hashing methods, the generation of image features is mostly performed once, in this work the weights of the network are updated again at each step. The objective function is $J_t^i\left(A_t^i, c_i\right)$ and the parameter θ_f is updated as in Equation 10.

$$\theta_f = \theta_f - \alpha_f \nabla_\theta J_t^i\left(A_t^i, c_i, \theta_f\right) \tag{10}$$

The goal of reinforcement learning is to find an optimal behavior strategy for the agent to obtain optimal rewards. This study aims to determine the optimization parameters for maximum reward.

$$\max G\left(\theta\right) = L_g(\theta) + R_g(\theta) \tag{11}$$

θ represents the model parameters, L_g is the reward of the action set and R_g is the reward of the objective function in the model. The model parameters are iteratively updated for the maximum reward.

In this work, hashing is considered as a dynamic Markov chain. The conditional probability distribution of the Markov chain is expressed as $p(s_{t+1}|s_t, a_t)$. $s_t \epsilon S$, $a_t \epsilon A$ and $s_{t+1} \epsilon S$; represent the state and action at time t and the next state, respectively. The objective is to learn the stochastic policy $\pi_\theta(a_t|s_t)$ conditional on the training parameter θ. A trajectory with t=1, 2,...,T

$$p_\theta(\tau) = p_\theta(s_1, a_1, s_2, a_2, \dots, s_\tau, a_\tau, s_{\tau+1}) == p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{12}$$

Let $g_t = G(a_t, s_{t+1})$ be the reward at time t. The objective function can find the value of θ^* that satisfies the optimal policy.

$$\theta^* = \arg max_\theta J(\theta) \tag{13}$$

Objective function $J(\theta)$,

$$J(\theta) = E_{\tau \sim p_\theta(\tau)}[\sum_{t=1}^{T} g_t] = E_{\tau \sim \pi_\theta(\tau)}[\sum_{t=1}^{T} g_t] \tag{14}$$

For the expected total reward of each set of actions taken, the Monte-Carlo Policy Gradient is used to optimize the model parameter. Gradient-based policy methods aim to model and optimize the policy directly. Policy gradient methods estimate the gradient in one iteration to maximize the expected total reward.

The policy gradient is expressed as $\nabla_\theta J(\theta) = \nabla_\theta E_{\tau\sim\pi_\theta(\tau)}[\sum_{t=1}^{T} g_t]$. In this study, Equation 15 is preferred among the many policy gradient definitions in the literature.

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)\, g(\tau)] \tag{15}$$

Expected total reward of the action set.

$$L_g(\theta) = E_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} g_t\right] == \sum_{t=1}^{T} \log[\pi_\theta(a_t|s_t)]\, g_t \tag{16}$$

s_t is the k th time in the loop in the RNN layer with the outputs of the RNN cell,

$$L_g(\theta) = \sum_{t=1}^{T}\sum_{i=1}^{k} \log\left[\mathrm{P}(a_{i,t}|\hat{s}_{i,t},\theta)\right] g_t \tag{17}$$

R_g is the global reward elated to the accuracy of all generated hash codes. Gradient descent algorithm is used to optimize the global reward. In model training we can calculate the sub-gradient for A_t^i, c_i is $\frac{\partial J_t^i\left(A_t^i, c_i\right)}{\partial\theta}$ by Equation 4.1. The hash codes generated with this training strategy are forced to be like those generated with Central Similarity.

The pseudo-code of the proposed method is roughly shown below.

Algorithm 2 Pseudo code of proposed method

```
Generate hash center of classes
While(1)
Create a history vector (hv) and assign all
zeros.
Generate image feature (fv) from CNN and
define state s=(fv,hv).
Generate actions from RNN. The input of RNN
is state (s).
Calculate the loss between the hash vector
and actions.
Calculate Backward-Propagation of Image
Adjust the learning rate every N epoch.
If loss < 0.001 break
End
```

3. Experiment Results

CIFAR10 and NUS-WIDE datasets, which are generally accepted in the literature, were used in this study. The CIFAR10 dataset contains about 60000 images of size 32 x 32, while NUS-WIDE consists of about 265000 images collected in Flicker and manually annotated with 81 concepts. For CIFAR10, we randomly select 1000 images as a query set and further randomly select 5000 images to form the training set. For NUS-WIDE datasets, we choose the 21 most prevalent concepts for our experimentation. Our query set consists of 2100 images, with 100 images per concept. Additionally, we randomly select an extra 10000 images for the training set.

The Hamming radius refers to the maximum number of bit positions by which two binary strings of equal length can differ from each other while still being considered within a specified distance from each other. In other words, it measures the extent of dissimilarity between two binary strings regarding the differing bit positions.

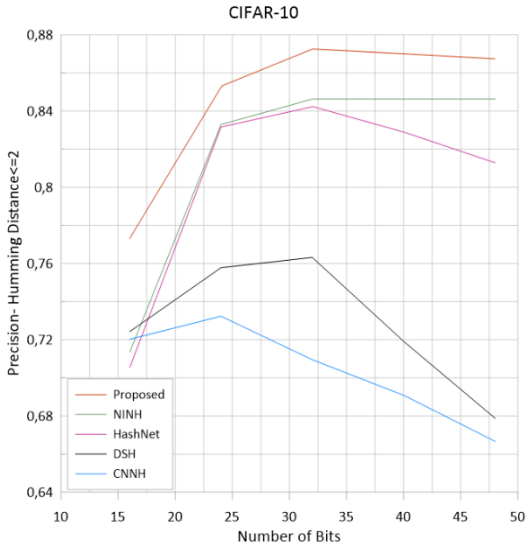

Figure 6 Precision within Hamming radius 2 using hash lookup for CIFAR Dataset

In this study, CNNH [28], DSH [31], HASHNET [9] and NINH [28], which are deep hashing approaches, were compared with the proposed method. When Figure 6 is analyzed for the images of the CIFAR10 dataset using Hamming Radius 2, the precision of the proposed method outperforms the other methods for all bit values. In addition, when the length of the hash codes increases, the number of images sharing the same Hamming code will decrease, so the image-matching process with Hamming radius 2 is unsuccessful for most algorithms. When Figure 6 is analyzed, it can be observed that the proposed method also performs better for longer hash codes.

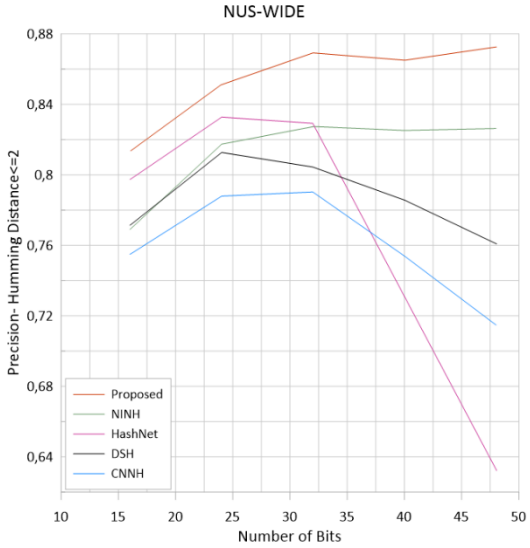

Figure 7 Precision within Hamming radius 2 using hash lookup for NUS-WIDE Dataset

In Figure 7, for NUS-WIDE, the proposed method outperforms the other methods at all hash code lengths.

The Mean Average Precision (MAP) is computed as the arithmetic mean of the Average Precisions (AP) obtained for each query within a given dataset. It is a metric commonly employed to assess the performance of information retrieval systems. It provides a single value that summarizes the overall performance of a retrieval system. Higher MAP values indicate better retrieval performance. As can be seen from Table 1, the MAP value is calculated for each hash code length. This value gives us for how good our model is.

$$AP = \frac{1}{R}\sum_{i=1}^{n}\frac{i}{Ri} \times rel_i \tag{16}$$

$$MAP = \frac{1}{N}\sum_{i=1}^{n} APi$$

R represents the overall count of true positive instances, n stands for the total number of images under consideration, APi denotes the precision at point i, and rel_i functions as a relevance indicator. The relevance function, characterized as an indicator function, assumes a value of 1 when the document at rank i is considered relevant, and it takes a value of 0 otherwise.

Table 1. MAP Values of Hash Bits

Methods	**CIFAR10**			**NUS-WIDE**		
	24 bit	32 bit	48 bit	24 bit	32 bit	48 bit
NINH	0.818	0.832	0.830	0.827	0.827	0.827
HashNet	0.823	0.840	0.843	0.833	0.830	0.840
DSH	0.712	0.751	0.720	0.804	0.815	0.800
CNNH	0.692	0.667	0.623	0.784	0.790	0.740
Proposed	0.857	0.868	0.865	0.859	0.862	0.865

In this study, MAP values of the proposed method and other methods are calculated for 24, 32 and 48 bit hash codes. MAP scores are calculated for hash codes of different lengths on the CIFAR10 dataset. The proposed method outperforms the other methods for all hash bit lengths. It is seen that the proposed method has the highest average MAP value of 0.863 and consistently outperforms the other methods. Similarly, for the NUS-WIDE dataset, When the MAP values are analyzed, the proposed method performs better at all hash bit lengths.

Conclusion

In this paper, we have proposed a Sequential and Correlated Image Hash Code Generation with Deep Reinforcement Learning (SCIHCGRL). To consider, the errors of previous hash functions in the deep reinforcement learning strategy, the RNN model is used, and a new model is presented that performs central similarity-based learning involving the correlation of hash functions. Thus, since the errors of the previous hash functions are considered during the sequential transfer of the images into binary codes, more efficient results are obtained in learning binary hash codes by adapting the Central Similarity metric to the model, which can achieve more successful results. Comparisons using CIFAR-10 and NUS-WIDE and datasets show that the proposed method gives better results.

Future work can be done to improve the time complexity of the algorithm. In addition, the proposed method can also be applied to videos by considering various properties of video media (correlation between video frames etc.).

References

[1] A. Swaminathan, Y. Mao and M. Wu, "Robust and secure image hashing," in IEEE Transactions on Information Forensics and Security, vol. 1, no. 2, pp. 215-230, June 2006, doi: 10.1109/TIFS.2006.873601.
[2] J. Wang, T. Zhang, N. Sebe, H. T. Shen et al., "A survey on learning to hash," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 769–790, 2018.
[3] S. Zhang, J. Li, M. Jiang, and B. Zhang, "Scalable discrete supervised multimedia hash learning with clustering," IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), vol. PP, no. 99, pp. 1–1, 2017.
[4] M. Kafai, K. Eshghi, and B. Bhanu, "Discrete cosine transform locality sensitive hashes for face retrieval," IEEE Transactions on Multimedia (TMM), vol. 16, no. 4, pp. 1090–1103, 2014.
[5] P. Li, M. Wang, J. Cheng, C. Xu, and H. Lu, "Spectral hashing with semantically consistent graph for image indexing," IEEE Transactions on Multimedia (TMM), vol. 15, no. 1, pp. 141–152, 2013.
[6] S. Zhang, J. Li, M. Jiang, and B. Zhang, "Scalable discrete supervised multimedia hash learning with clustering," IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), vol. PP, no. 99, pp. 1–1, 2017.
[7] V. E. Liong, J. Lu, Y.-P. Tan, and J. Zhou, "Deep video hashing," IEEE Transactions on Multimedia (TMM), 2016.
[8] K. Ding, B. Fan, C. Huo, S. Xiang, and C. Pan, "Cross-modal hashing via rank-order preserving," IEEE Transactions on Multimedia (TMM), vol. 19, no. 3, pp. 571–585, 2017.
[9] Z. Cao, M. Long, J. Wang, and S. Y. Philip. Hashnet: Deep learning to hash by continuation. In ICCV, pages 5609–5618, 2017.
[10] W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. arXiv preprintrXiv:1511.03855, 2015.

[11] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov. Hamming distance metric learning. In Advances in neural information processing systems, pages 1061–1069, 2012.
[12] Yuan, L., Wang, T., Zhang, X., Tay, F. E., Jie, Z., Liu, W., & Feng, J. (2020). Central similarity quantization for efficient image and video retrieval. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 3083-3092).
[13] Y. Li Deep reinforcement learning: An overview,2017
[14] Y. Peng, J. Zhang and Z. Ye, "Deep Reinforcement Learning for Image Hashing," in IEEE Transactions on Multimedia, vol. 22, no. 8, pp. 2061-2073, Aug. 2020, doi: 10.1109/TMM.2019.2951462.
[15] P. Indyk and R. Motwani. Approximate Nearest Neighbor {Towards Removing the Curse of Dimensionality. In Proceedings of the 30th Symposium on Theory of Computing, 1998, pp. 604{613.
[16] A. Gionis, P. Indyk, R. Motwani et al., "Similarity search in high dimensions via hashing," in International Conference on Very Large Data Bases (VLDB), vol. 99, no. 6, 1999, pp. 518–529.
[17] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in Annual Conference on Neural Information Processing Systems (NIPS), 2009, pp. 1509–1517.
[18] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in Annual Conference on Neural Information Processing Systems (NIPS), 2009, pp. 1753–1760.
[19] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 817–824.
[20] G. Irie, Z. Li, X.-M. Wu, and S.-F. Chang, "Locally linear hashing for extracting non-linear manifolds," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 2115–2122.
[21] Liu, W.; Mu, C.; Kumar, S.; and Chang, S. 2014. Discrete graph hashing. In NIPS, 3419–3427.
[22] Jiang, Q.-Y., and Li, W.-J. 2015. Scalable graph hashing with feature transformation. In IJCAI, 2248–2254.
[23] Liu, H.; Ji, R.; Wu, Y.; and Liu, W. 2016b. Towards optimal binary code learning via ordinal embedding. In AAAI, 1258–1265.
[24] X. Shen et al., "Semi-paired discrete hashing: Learning latent hash codesfor semi-paired cross-viewretrieval," IEEE Trans. Cybern., vol. 47, no. 12,pp. 4275–4288, Dec. 2017.
[25] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen, ``Unsupervised deep hashing with similarity-adaptive and discrete optimization," IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 12, pp. 30343044, Dec. 2018.
[26] X. Shen et al., "Multiview discrete hashing for scalable multimedia search," ACM Trans. Intell. Syst. Technol., vol. 9, no. 5, pp. 1–21, 2018.
[27] Kong, W., and Li, W.-J. 2012. Isotropic hashing. In NIPS,1655–1663.
[28] 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.
[29] H. Lai,Y. Pan,Y. Liu, and S.Yan, "Simultaneous feature learning and hash coding with deep neural networks," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., 2015, pp. 3270–3278.
[30] 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.
[31] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2064–2072.
[32] T. Yao, F. Long, T. Mei, and Y. Rui, "Deep semantic-preserving and ranking-based hashing for image retrieval," in Proc. Int. Joint Conf. Artif. Intell., 2016, pp. 3931–3937.
[33] 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.
[34] Wu, L., Ling, H., Li, P., Chen, J., Fang, Y., & Zhou, F. (2019). Deep supervised hashing based on stable distribution. IEEE Access, 7, 36489-36499.
[35] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," Journal of artificial intelligence research, vol. 4, pp. 237–285, 1996.

Conflict of Interest Notice
The authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval and Informed Consent
It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of data and material
Not applicable.

Plagiarism Statement
This article has been scanned by iThenticate ™.