Two-Stage Sequential Losses based Automatic Hash Code Generation using Siamese Network

Şaban Öztürk[1*]

[1] Amasya University, Technology Faculty, Electrical and Electronics Engineering Department, Amasya, Turkey (ORCID: 0000-0003-2371-8173)

(1st International Conference on Computer, Electrical and Electronic Sciences ICCEES 2020 – 8-10 October 2020)

(**DOI:** 10.31590/ejosat.801927)

ATIF/REFERENCE: Öztürk, Ş. (2020). Two-Stage Sequential Losses based Automatic Hash Code Generation using Siamese Network. *European Journal of Science and Technology*, (Special Issue), 39-46.

Abstract

Today, large-scale image retrieval methods are used for fast access to increasing visual information. Hashing methods are an image retrieval approach that provides computationally effective and high-speed access. Hash codes created with hand-crafted features extracted from images in the past are now built with deep learning architectures. Convolutional neural network (CNN)'s an impressive performance using optimized features strengthens this trend day by day. However, it contradicts the hash codes consisting of '0' and '1' values and the procedure of updating the parameters and output of CNN architectures. To solve this problem, solutions such as bringing the bits in the CNN output to '0' and '1' values with a threshold value, and converting the output bits to binary values with the help of loss functions in the CNN output are presented in the literature. It is very useful for studies involving loss function solutions for end-to-end training. However, in these studies, loss weights are generally empirical. In order to solve this situation, a framework consisting of two steps is suggested in this study. In the first stage, Euclidean distance is used in the output of a CNN architecture that is trained in a pairwise manner. After reducing the distance between the two images below the specified value, in the second step, the network parameters are transferred, and the output is drawn to binary values with binarization loss. Thus, binary hash codes are obtained automatically for each image without using any additional weight.

Keywords: CNN, Content-based image retrieval, CBIR, medical image retrieval, hashing, binarization loss, Euclidean distance.

Siamese Network kullanarak İki Aşamalı Sıralı Kayıplara dayalı Otomatik Hash Kodu Üretimi

Öz

Günümüzde, artan görsel bilgiye hızlı erişim için büyük ölçekli görüntü erişimi yöntemleri kullanılmaktadır. Hashing yöntemleri, hesaplama açısından etkili ve çok hızlı erişim sağlayan görüntü erişimi yaklaşımlarındandır. Geçmişte görüntülerden çıkarılan el yapımı özelliklerle oluşturulan karma kodlar artık derin öğrenme mimarileriyle oluşturulmaktadır. Evrişimli sinir ağının (CNN) optimize edilmiş özellikleri kullanan etkileyici performansı, bu eğilimi her geçen gün güçlendirmektedir. Ancak, '0' ve '1' değerlerinden oluşan karma kodlarla ve CNN mimarilerinin parametrelerini ve çıktılarını güncelleme prosedürü ile çelişir. Bu problemi çözmek için CNN çıktısındaki bitlerin eşik değeri ile '0' ve '1' değerlerine getirilmesi ve CNN çıkışındaki kayıp fonksiyonları yardımıyla çıkış bitlerinin ikili değerlere çevrilmesi gibi çözümler bulunmaktadır. Uçtan uca eğitim için kayıp fonksiyonu çözümlerini içeren çalışmalar için çok kullanışlıdır. Ancak bu çalışmalarda kayıp ağırlıkları genellikle ampiriktir. Bu durumu çözmek için bu çalışmada iki aşamadan oluşan bir çerçeve önerilmektedir. İlk aşamada, çift olarak eğitilmiş bir CNN mimarisinin çıktısında Öklid mesafesi kullanılır. İki görüntü arasındaki mesafe belirtilen değerin altına indirildikten sonra ikinci adımda ağ parametreleri aktarılır ve çıktı ikili değerlere çekilir. Böylelikle herhangi bir ek ağırlık kullanılmadan her görüntü için ikili hash kodları otomatik olarak elde edilir.

Anahtar Kelimeler: CNN, İçerik tabanlı görüntü erişimi, CBIR, tıbbi görüntü erişimi, hashing, ikilileştirme kaybı, Öklid mesafesi.

[*] Şaban Öztürk, Amasya University, Technology Faculty, Electrical and Electronics Engineering Department, Amasya, Turkey, ORCID: 0000-0003-2371-8173, saban.ozturk@amasya.edu.tr

1. Introduction

One of the consequences of the ease of access to imaging devices and the proliferation of the internet is the large number of images. Today, the number of accessible images continues to increase rapidly. Finding the desired content among these image data is a very challenging problem. Many researchers turn their attention to this direction, especially to produce fast and effective solutions [1].

Content-based image retrieval (CBIR) methods find the most similar images from a large image dataset in response to a query image. Although it seems ideal to use the nearest neighbor (NN) search for this, such a search process can take forever [2]. Hashing-based solutions are suggested to overcome this problem. Hashing methods generally consist of functions expressing high-dimensional data as low-dimensional binary codes. In this way, both computational complexities are reduced and the access process is very fast. Hashing methods in the literature are generally divided into two as data-independent and data-dependent. Since the hashing methods produce binary codes according to the image features, a distinction can be made today as hand-crafted and CNN methods. In this study, firstly, hand-crafted and CNN methods are examined in terms of extracting the features. Then, data-dependent and data-independent methods are examined for hash code generation.

Features extracted by hand-crafted methods usually require experience, and success may be reduced for data in different domains. However, these methods are highly preferred before CNN. SIFT [3], texture features [4], GLCM [5] and multiple features [6] methods are among the hashing studies performed with hand-crafted features. Many hand-crafted methods are not designed for retrieval tasks, so they cannot close the semantic gap. For this reason, CNN-based hashing methods are preferred today. CNN approaches help fill the semantic gap by automatically learning problem-based features. Multi-level deep features [7], capsule network vectors [8], and joint deep network [9] are some of the outstanding studies of CNN architectures.

The hash code generation part is examined in two categories as data-dependent and data-independent. Data-independent methods create random hash codes. These hash codes are produced independently of the dataset and without training. Locality sensitive hashing (LSH) [10] is the most well-known data-independent hashing method. This approach requires quite long hash codes. For this reason, it is not seen as an effective approach today. Data-dependent hashing methods calculate the distance between training data. It is divided into three groups as unsupervised, semi-supervised, and supervised hashing. Spectral hashing [11] and iterative quantization [12] are among the most well-known unsupervised approaches. Semi-Supervised Adversarial Deep Hashing (SSAH) [13] and Anchor-Based Self-Ensembling [14] are the high-performance semi-supervised studies that have been proposed recently. Supervised hashing methods are the most widely studied hashing method today [15, 16]. It produces the most successful results with a sufficient number of labeled data and correct architecture.

Pointwise, pairwise, and tripletwise (multi-wise) samples approaches are used in the training of hashing methods. Pointwise based methods use only one image as input. These methods generally aim to generate hash code by classifier or regression analysis. For pairwise based training, two images are used as inputs. These images can be similar or dissimilar. Tripletwise based methods use three images as input. One image is a query image, another is similar and the other is dissimilar images.

In this study, the input images are used as a pairwise manner. Because it can produce more effective results than pointwise manner input. Also, it produces results faster than triplet manner inputs. When the CBIR studies in the literature are examined, it is seen that end-to-end manner training is preferred. However, there are some difficulties in generating hash codes with your end-to-end training. The most important of these is the loss function and gradients. If the loss is calculated with a distance function such as Euclidean, the CNN output cannot automatically generate binary values. If a loss function is used in the hamming space, the CNN output can generate '0' and '1'. But in this case, updating the CNN parameters becomes a much bigger challenge. For this reason, the process of converting CNN output to binary codes with a threshold or margin value is used in the literature. Another and more effective method uses more than one loss function. In such a case, the loss function can be used together for many tasks such as measuring distance, converting the output to binary codes, and maintaining balance. This is the current approach in the literature. However, the most significant disadvantage of this method is that weight is assigned to each loss function. Calculating the values of these weights automatically creates a sizable computational complexity. It also requires fundamental changes in popular CNN architectures. For this reason, researchers determine these values by using empirical values or light education. In this study, a two-stage framework is proposed to overcome this difficulty. The testing phase of the proposed method, which has two stages of training, is an end-to-end manner. In the first stage, a CNN architecture is trained in a pairwise manner with only Euclidean distance. The parameters of the network are updated by determining the distance according to whether the pairwise images are similar or dissimilar. In the second stage, the binarization loss function is added by changing the loss function at the exit of the network trained in the first stage. In this case, a pairwise training process is performed again, with less iteration. The network, which has already learned the distance between images, learns to generate binary codes in the second stage. Finally, according to margin values, all values are made absolute '0' and absolute '1'.

The rest of this paper is organized as follows. Section 2 provides details and parameters of the proposed framework. Experimental results are presented in Section 3. Section 4 includes discussion and future directions. Finally, the conclusion is given in Section 5.

2. Material and Method

2.1. Datasets

Two datasets are used in this study to test the performance of the proposed framework. Having two datasets in different domains is very useful for fair measurement of the performance of the proposed method. For this reason, the first dataset consists of general objects. The second dataset contains medical images.

Dataset 1: The SIVAL (Spatially Independent, Variable Area, and Lighting) benchmark is used as the first dataset (obtained from https://www.cs.wustl.edu/~sg/accio/SIVAL.html) This dataset contains 25 different categories of objects. Each category consists of 60 images. Each image contains a highly diverse background. Objects in the images can be anywhere in the view. They could also be photographed from a distance or close up, photographed from a wide-angle, or part of it may not be visible. The resolution of images is 1024 by 768 pixels. All images are in RGB color space. Figure 1 (a) shows some sample images from this dataset.

Dataset 2: NEMA CT (obtained from http://ftp//medical.nema.org/medical/Dicom/Multiframe/) is selected as a medical dataset. The images in this data are created in the DICOM format provided by the National Electrical Manufacturers Association. In this study, images belonging to 10 classes consisting of 663 images are used. The resolution of the images is 512x512 pixels. All images are gray levels. Figure 1 (b) shows some sample images from this dataset.

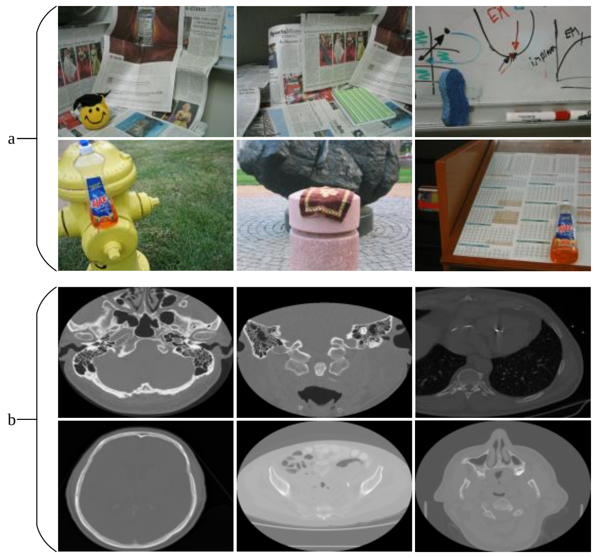

Fig. 1 Some samples from datasets, a) Samples from the first dataset, b) Samples from the second dataset

2.2. Proposed Method

In order to better understand the proposed framework, it is explained in three sections. The first of these is the CNN architecture and operation used. The second is the Euclidean distance trained network parameters section. In the third part, training with binarization loss is explained. The proposed framework is shown in Figure 2.

CNN architectures are very successful in solving image processing problems thanks to the layers they have. For this reason, they are encountered in almost all image processing works today [17]. Almost every CNN architecture contains several primary layers. The new architectures produced today are realized by additions to these primary layers or by proposing new layers. Although there are many comprehensive resources on CNN architectures, in this study, the layers used in the proposed framework will be discussed quickly to create a brief background. The convolution layer is the most basic layer of CNN architecture. This layer, which is a two-dimensional kernel, learns the features of images. Convolution kernels are subjected to 2D convolution operation with the image by moving them over the images. This process, called parameter sharing, considerably reduces the number of parameters. Rectified linear unit (ReLU) disrupts the linearity of the network, using *y=max(0,x)*. Another layer that serves to reduce the total number of parameters in the network is the pooling layer. This layer is used after the convolution layer for increasing efficiency. It is to preserve important information in one tensor and transfer it to the next layer. There are various types, but max-pooling is preferred in this study. In the max-pooling process, only the pixel with the maximum value among the pixels under the pooling window is transferred to the next layer. A fully connected layer (FCL) is a type of multi-layered perceptron (MLP) structure. A CNN network consisting of only convolution, ReLU, and max-pooling layers can be calculated using Equation 1.

$$L_{Next} = pool_{n\times n}^{\max}\left(\sigma\left(w\otimes I+b\right)\right) \tag{1}$$

where L_{Next} represents the input of the next layer or output of the current layer, *pool* is the max-pooling operator with n by n windows size, σ represents ReLU activation function, w represents weights of the convolution layer, I represents the input of the convolution layer, b is the bias value. Table 1 shows layers of the proposed CNN architecture.

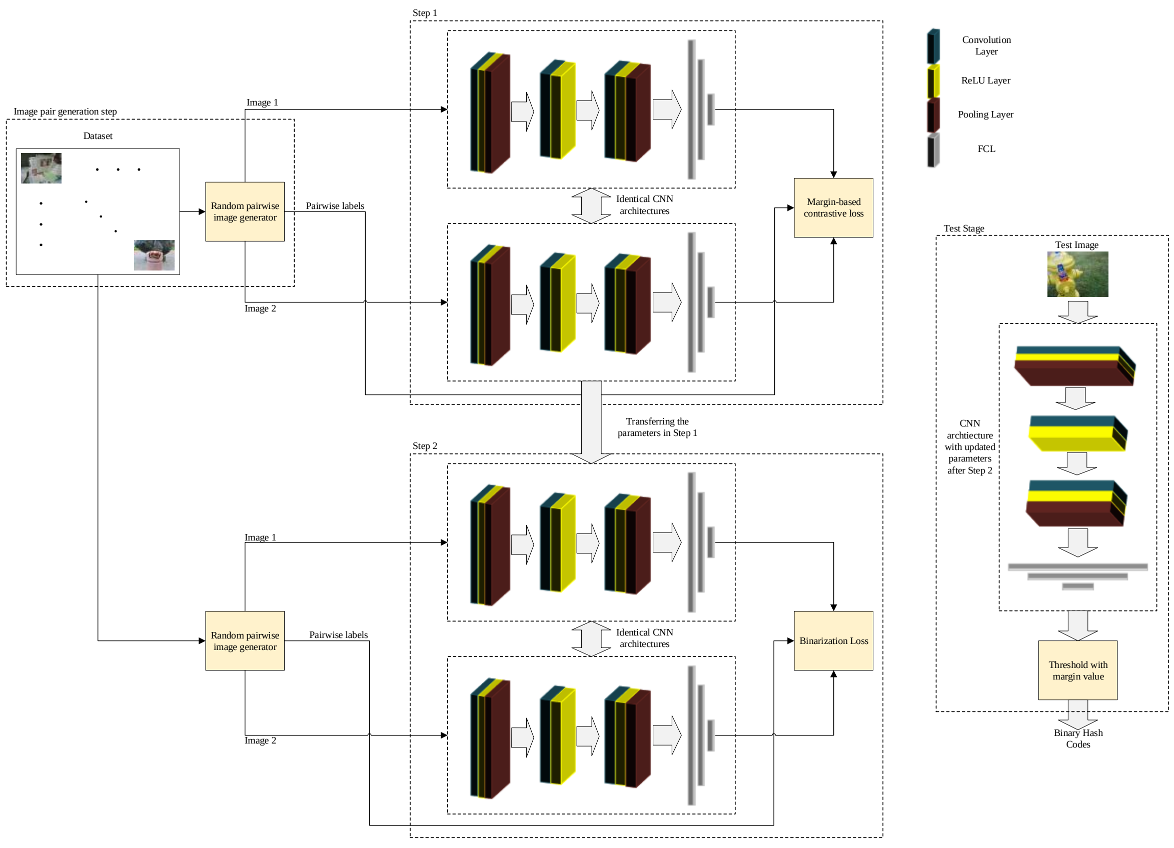
Step 1
Image pair generation step
Dataset
Image 1
Random pairwise image generator
Pairwise labels
Image 2
Identical CNN architectures
Margin-based contrastive loss
Convolution Layer
ReLU Layer
Pooling Layer
FCL
Test Stage
Test Image
CNN archtiecture with updated parameters after Step 2
Threshold with margin value
Binary Hash Codes
Transferring the parameters in Step 1
Step 2
Image 1
Random pairwise image generator
Pairwise labels
Image 2
Identical CNN architectures
Binarization Loss

Fig. 2 The Proposed Framework

In the first stage of the proposed framework, image pairs are generated randomly from the images in the data. If two images are similar, the label value is assigned '1', otherwise '0'. Let's say I_p^N for image pairs L^N, N is the number of image pairs, when $I_p^N=\{I_1, I_2\}$, $L^N \epsilon \{0,1\}^N$. After creating pairs of images and pairwise labels, training is performed using contrastive loss in Step 1.

Table 1. Configurations of the proposed CNN architecture

Type	Filter size/stride	Type	Filter size/stride
Input	128x128x3	Convolution 3	3x3x128x128/1
Convolution 1	3x3x3x64/1	ReLU 3	-
ReLU 1	-	Max-Pooling 2	2x2/1
Max-Pooling 1	2x2/2	FCL 1	4096
Convolution 2	3x3x64x128/1	FCL 2	1024
ReLU 2	-	FCL 3	d

The aim in the first step is to measure the similarity between image pairs and optimize the network accordingly. Using the Euclidean space for this is one of the least costly choices. Placing similar image features close to each other in Euclidean space and disparate features to far corners of space helps to solve the problem. Margin-based contrastive loss (MBCL) function in [18] is used to implement Step 1. MBCL is calculated in Equation 2.

$$MBCL = \frac{1}{2} LD^2 + \frac{1}{2}(1-L)\left\{\max\left(0, m-D\right)\right\}^2 \qquad (2)$$
$$\text{where } D = \left\| f(I_1) - f(I_2) \right\|_2$$

in which L is pairwise labels (If I_1 and I_2 are similar, L gets '1', if I_1 and I_2 are dissimilar, L gets '0'.), $f(I_1)$, and $f(I_2)$ are feature vectors of I_1 and I_2. m represents the margin threshold value.

The purpose of step 1 in the proposed framework is to determine the similarities between image pairs. The parameters of the Siamese network used for this are updated according to Euclidean distance. As a result of Step 1, a CNN network with updated weights is created. However, CNN architecture as such is not suitable for generating hash code. In the second step, the network and its parameters trained during the first step are used. Only the loss function is changed. In this step, the number of iterations is less than step 1. Because in step 1, similarities are learned strongly. In the second step, only these values are binarized. If step 2 is expressed in $b=h(d)$, then h represents the hashing function and d represents the size of the features. In this case, we aim to represent the feature vector as low-dimensional (k-dimensional) and binary, $b\in\{0,1\}^k$. The loss function suggested in this section pushes the last layer activations to binary values and ensures a balanced distribution of 0 and 1. Equation 3 is used to update the last layer activations of the proposed CNN inspired by [19].

$$L_{Bin} = \sum_{i=1}^{J} \left\| f\left(I_{current} - m\right) \right\|^2 - \frac{mean\left(f\left(I_{current}\right)\right)}{2} \tag{3}$$

The length of the hash code obtained at the output of Step 2 is the same as the last layer of the CNN architecture. The code generated is not a full binary code, it consists of numbers close to '0' and '1' values. In the test phase, the margin-threshold function is added to the end of the network and the generated code is converted into full binary.

3. Results and Discussion

3.1. Experimental Results

The entry of the proposed CNN architecture is designed for 128x128 pixel dimensions. For this reason, the dimensions of all the images in both data are resized to 128x128 pixels. However, the images in the first data are colored, and the images in the medical data are gray levels. The only architectural change between the two experiments is made for this section. The input layer for the first dataset is 128x128x3. The input layer for the second dataset is 128x128x1.

Pairwise training is used in two steps of the proposed framework. For this purpose, 400 image pairs are randomly selected from the images in the dataset at each iteration. These image pairs are automatically tagged to be similar and dissimilar. The margin threshold (m) value is selected as 0.3. The learning rate is used as 10^{-5}. The gradient decay is 0.9. Adam optimizer is used to optimize the parameters of the proposed CNN architecture. To evaluate the performance of the proposed method, the output of the network in Step 1 is examined first. For this, the output neurons of this network are changed to 8, 16, and 32, respectively. Figure 3 shows the outputs in the first step of the proposed Siamese network. The curves in the upper row are related to dataset 1, and the curves in the lower row are related to the medical dataset. As can be seen from Figure 3, as the number of bits increases, the representation power of the network increases. However, as the number of parameters to be trained increases, time and computational complexity also increase.

The other information Figure 3 shows is that datasets with fewer classes can be learned faster. Because the medical dataset contains fewer classes, it converges faster. On the other hand, the binary codes generated by Step 2 are more essential for this study. For this reason, the output curves of Step 2 are analyzed in Figure 4.

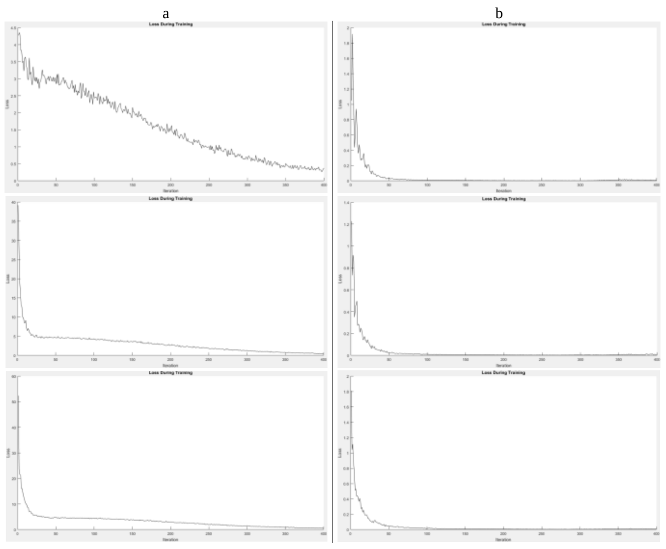

Fig. 3 Training curves of Step 1, a) Dataset 1, b) Medical dataset

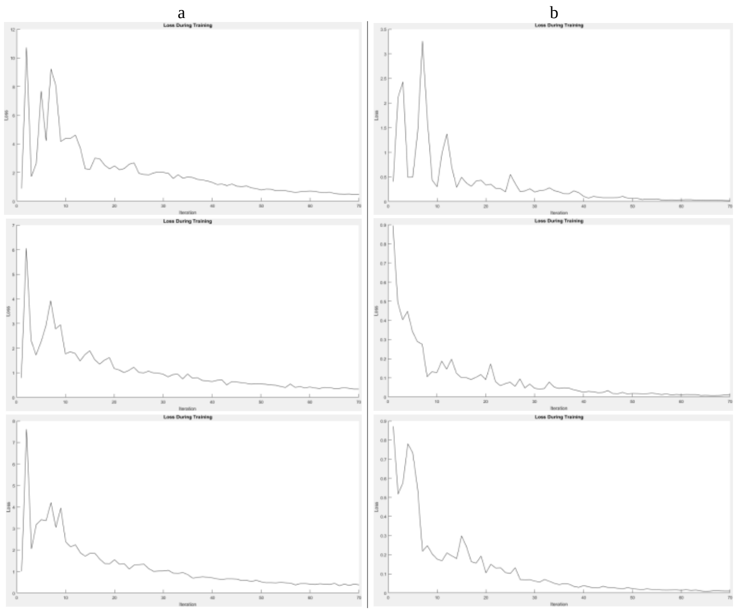

Fig. 4 Training curves of Step 2, a) Dataset 1, b) Medical dataset

3.2. Discussion

The proposed framework can be easily used with other CNN architectures. However, as the depth of the architecture increases, higher-capacity hardware is needed. In its current form, the proposed architecture performs as well as many state-of-the-art methods. If we examine the experiments in order, it turns out that the retrieval performances were satisfactory in two experiments. Because the number of classes is high in Experiment 1, using an 8-bit hash code produces fast results, but precision is insufficient. Sample retrieval images obtained from dataset 1 with an 8-bit hash code are shown in Figure 5.

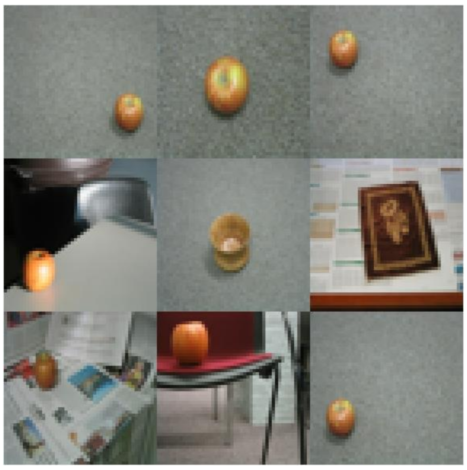

Fig. 5 Sample retrieval images obtained from dataset 1 with 8-bit hash code

Increasing the number of bits of the hash code increases the precision value, but also increases the retrieval time. The results produced by the 16-bit hash code for Dataset 1 are quite satisfactory. Besides, the results produced by the 32-bit hash code are relatively high. However, it is better to use a 16-bit hash code for dataset 1 for efficiency. Sample retrieval images obtained from dataset 1 with 16-bit hash code are shown in Figure 6.

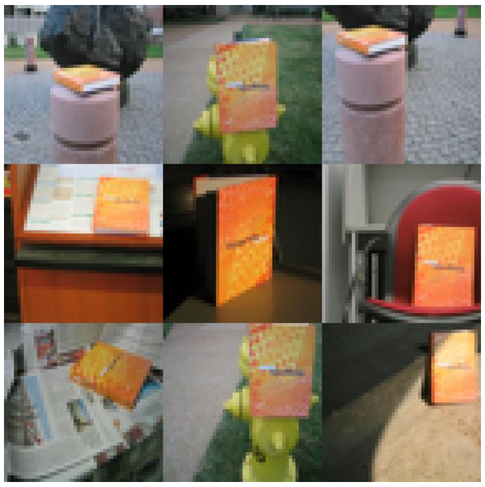

Fig. 6 Sample retrieval images obtained from dataset 1 with 16-bit hash code

Setting the medical dataset class number as 10 enables hash codes with less bit count to produce high precision. For this reason, an 8-bit hash code has both satisfactory precision and fast speed. It is appropriate to choose it in terms of efficiency. Sample retrieval images obtained from dataset 2 with an 8-bit hash code are shown in Figure 7.

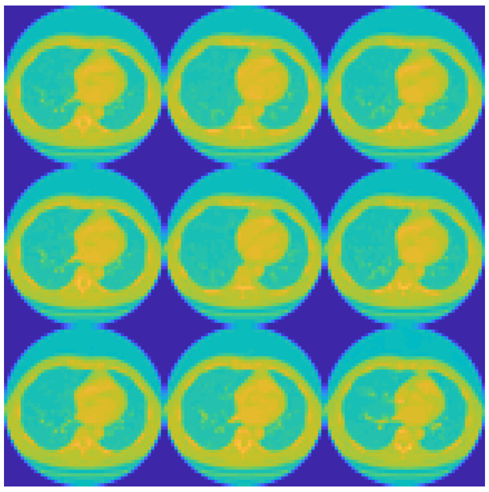

Fig. 7 Sample retrieval images obtained from dataset 2 with 8-bit hash code

4. Conclusions and Recommendations

In this study, a deep hashing method that includes the use of multiple loss functions with serial deep networks is introduced. The proposed method consists of combining margin-based contrastive loss and binarization-driven loss functions in series. In fact, there are studies with similar loss functions, but such studies usually use a single network. It combines all loss functions by assigning a separate weight. Thus, the network is updated with a single value. However, it is a big problem to calculate the importance value for each loss in such approaches. The proposed framework overcomes this problem quite simply. The proposed sequential Siamese method does not require loss weights. Also, the testing phase of the proposed framework that is not the training phase end-to-end is end-to-end. In addition, the training phase of the proposed framework is not end-to-end. But the testing phase is end-to-end. Compared to other state-of-the-art methods, the performance of the proposed method is higher than most. According to architectures with similar performance, its retrieval time is almost twice as efficient. Future studies will focus on end-to-end sequential architectures.

Acknowledge

Conflict of interest: The authors declare that they have no conflicts of interest.

Human and animal rights: The paper does not contain any studies with human participants or animals performed by any of the authors.

This research is funded by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant number 120E018.

References

[1] Ashraf, R., Ahmed, M., Ahmad, U., Habib, M. A., Jabbar, S., & Naseer, K. (2020). MDCBIR-MF: multimedia data for content-based image retrieval by using multiple features. Multimedia tools and applications, 79(13), 8553-8579.

[2] Öztürk, Ş. (2020). Stacked auto-encoder based tagging with deep features for content-based medical image retrieval. Expert Systems with Applications, 161, 113693.

[3] Demir, B., & Bruzzone, L. (2015). Hashing-based scalable remote sensing image search and retrieval in large archives. IEEE transactions on geoscience and remote sensing, 54(2), 892-904.

[4] Alsmadi, M. K. (2020). Content-Based Image Retrieval Using Color, Shape and Texture Descriptors and Features. Arabian Journal for Science and Engineering, 1-14.

[5] Mukherjee, A., & Gaurav, K. (2016). Content based image retrieval using GLCM. International Journal of Innovative Research in Computer and Communication Engineering, 4(11), 20142-20149.

[6] Liu, X., He, J., & Lang, B. (2014). Multiple feature kernel hashing for large-scale visual search. Pattern Recognition, 47(2), 748-757.

[7] Ng, W. W., Li, J., Tian, X., Wang, H., Kwong, S., & Wallace, J. (2020). Multi-level supervised hashing with deep features for efficient image retrieval. Neurocomputing.

[8] Li, Y., Zhang, R., Miao, Z., & Wang, J. (2019, October). CapsHash: Deep Supervised Hashing with Capsule Network. In 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP) (pp. 1-5). IEEE.

[9] Chen, Y., Lu, X., & Li, X. (2020). Supervised deep hashing with a joint deep network. Pattern Recognition, 105, 107368.

[10] Gionis, A., Indyk, P., & Motwani, R. (1999, September). Similarity search in high dimensions via hashing. In Vldb (Vol. 99, No. 6, pp. 518-529).

[11] Weiss, Y., Torralba, A., & Fergus, R. (2009). Spectral hashing. In Advances in neural information processing systems (pp. 1753-1760).

[12] Gong, Y., Lazebnik, S., Gordo, A., & Perronnin, F. (2012). Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE transactions on pattern analysis and machine intelligence, 35(12), 2916-2929.

[13] Jin, S., Zhou, S., Liu, Y., Chen, C., Sun, X., Yao, H., & Hua, X. S. (2020). SSAH: Semi-Supervised Adversarial Deep Hashing with Self-Paced Hard Sample Generation. In AAAI (pp. 11157-11164).

[14] Shi, X., Guo, Z., Xing, F., Liang, Y., & Yang, L. (2020). Anchor-Based Self-Ensembling for Semi-Supervised Deep Pairwise Hashing. International Journal of Computer Vision, 1-18.

[15] Luo, Y., Yang, Y., Shen, F., Huang, Z., Zhou, P., & Shen, H. T. (2018). Robust discrete code modeling for supervised hashing. Pattern Recognition, 75, 128-135.

[16] Yao, T., Han, Y., Wang, R., Kong, X., Yan, L., Fu, H., & Tian, Q. (2020). Efficient discrete supervised hashing for large-scale cross-modal retrieval. Neurocomputing, 385, 358-367.

[17] Öztürk, Ş., & Akdemir, B. (2019). A convolutional neural network model for semantic segmentation of mitotic events in microscopy images. Neural Computing and Applications, 31(8), 3719-3728.

[18] Hadsell, R., Chopra, S., & LeCun, Y. (2006, June). Dimensionality reduction by learning an invariant mapping. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) (Vol. 2, pp. 1735-1742). IEEE.

[19] Roy, S., Sangineto, E., Demir, B., & Sebe, N. (2020). Metric-Learning-Based Deep Hashing Network for Content-Based Retrieval of Remote Sensing Images. IEEE Geoscience and Remote Sensing Letters.