Optimized Binary Hashing Codes Generated by Siamese Neural Networks for Image Retrieval

Abin Jose, Timo Horstmann and Jens-Rainer Ohm
Institut für Nachrichtentechnik
RWTH Aachen University
Aachen, Germany
jose@ient.rwth-aachen.de

***Abstract*—In this paper, we use a Siamese Neural Network based hashing method for generating binary codes with certain properties. The training architecture takes a pair of images as input. The loss function trains the network so that similar images are mapped to similar binary codes and dissimilar images to different binary codes. We add additional constraints in form of loss functions that enforce certain properties on the binary codes. The main motivation of incorporating the first constraint is maximization of entropy by generating binary codes with the same number of 1s and 0s. The second constraint minimizes the mutual information between binary codes by generating orthogonal binary codes for dissimilar images. For this, we introduce orthogonality criterion for binary codes consisting of the binary values 0 and 1. Furthermore, we evaluate the properties such as mutual information and entropy of the binary codes generated with the additional constraints. We also analyze the influence of different bit sizes on those properties. The retrieval performance is evaluated by measuring Mean Average Precision (MAP) values and the results are compared with other state-of-the-art approaches.**

***Index Terms*—Siamese Neural Networks, Binary Hashing, Image Retrieval, Code Property Training, Information Theoretic Criteria.**

I. Introduction

Thousands of images are uploaded to the internet every day. In order to effectively search through the vast number of images, fast image retrieval is necessary. One approach for fast image retrieval is hashing, which maps the high-dimensional data onto a compact binary code [1]. The performance of hashing methods heavily depends on the chosen hashing function. With the development of Convolutional Neural Networks (CNNs), CNN-based hashing methods have gained popularity. In the recent years, CNNs achieved best results in several computer vision tasks such as image classification [2].

A. Related work

Hashing has become an important step for making efficient storage and retrieval of images and videos. Already existing hashing methods are divided into mainly data dependent and data independent approaches. Locality Sensitive Hashing (LSH) [3] uses random linear projections to generate optimum binary codes. Super-bit LSH [4] and non-metric LSH [5] are two popular recently proposed variants. Data dependent methods effectively utilize the training data while learning the binary codes and are classified into supervised and unsupervised approaches. Iterative quantization (ITQ) [6] is a popular unsupervised approach. Popular supervised models are Supervised Hashing with kernels [7], Order Preserving Hashing [8], and Supervised Discrete Hashing [9] which uses the label information of training data into account. Convolutional Neural Network Hashing (CNNH) [10] is one of the initial attempt to include deep learning for hashing. [11] proposes a triplet ranking based approach for hashing. Another approach of binary hashing is using a Siamese Neural Network (SNN) [12] which learns the similarity between the images. This means similar images are mapped to similar binary codes and dissimilar images to distinguishable binary codes. This similarity preserving characteristic is essential for image retrieval. When a retrieval system receives a query image, it calculates the corresponding binary code. Then it searches through the database for similar images. The similarity of images can simply be determined by calculating the Hamming distance of the binary codes. SNN performs well in tasks such as similarity comparison of images [13]. In addition to the similarity preserving constraint of the binary code, more constraints are often enforced upon the binary codes in order to generate the binary codes with some additional properties. These include the maximization of the entropy [14], [15] or generating binary codes that have independent bits as proposed in [16] and [17].

In this paper, we present a supervised SNN-based hashing method, which maps images onto compact binary codes. We added 2 constraints to the network in order to improve the properties of the binary codes. The first constraint maximizes the entropy of the bits by generating balanced binary codes that have the same number of 1s and 0s as proposed by [15]. The second constraint is responsible for generating balanced and orthogonal binary codes. For this, we introduced a novel approach for the orthogonality of binary codes consisting of the values 0 and 1. This constraint minimizes the mutual information between the bits and generates binary codes with independent bits.

Another major contribution of our work is the optimum selection of margin m for the hinge embedding loss which is a criterion used for training SNN for learning image similarity. Furthermore, there has not been much research aimed at measuring the actual entropy and mutual information of the binary codes to the best of our knowledge. Unlike most other papers that added similar constraints to generate binary codes, we evaluated the actual influence of the constraints on the properties of the binary codes with variation of the bit sizes. The rest of the paper is organized as follows: Section II discusses the neural network architecture and loss criteria. In section III, experimental results are analyzed. Concluding remarks are drawn in section IV.

II. Network architecture and loss functions

The SNN used for training is shown in Fig. 1. For each training step, the image pairs are generated first. The batch size n is set to 64, and thus 64 images are selected out of the training dataset. These images are called anchor images. For each anchor image one image belonging to the same class (positive pair) and one belonging to a different class (negative pair) is selected. The details of online generation of training pairs is explained in our previous work [18]. The sub-networks of SNN consists of CNNs followed by a sigmoid

ISBN 978-90-827970-1-5 © EURASIP 2018

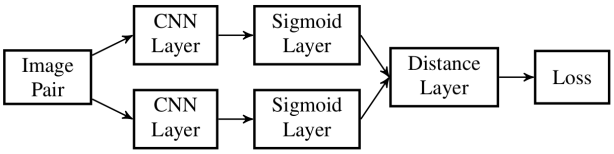

Fig. 1: Siamese neural network architecture.

layer. The CNNs extract features from the images. The sigmoid layer squashes the elements of the feature vectors in the range of $\{0, 1\}$. As the network is trained, the elements assume values close to 0 or 1. b is the number of bits of the binary vectors. In each training step, the network first receives the positive pair and then the negative pair. For each case, the loss is calculated and the weights are updated separately. The network is trained using stochastic gradient descent. The different training criteria are discussed in the subsequent sections.

A. Hinge Embedding Criterion

The hinge embedding criterion (HE) is the criterion used to train the network so that each class gets its own unique binary code. It receives the distance vector ($\boldsymbol{d}$) between image pairs and the corresponding labels y indicating similar or dissimilar pairs. This loss function minimizes the distance between similar pairs and increase the distance between dissimilar pairs such that it is higher than the margin m. The loss is calculated as follows:

$$L_{\mathrm{he}}(\boldsymbol{d}, y) = (1 - y)L_{\mathrm{sim}}(\boldsymbol{d}) + yL_{\mathrm{dis}}(\boldsymbol{d}).$$

Label $y = 1$ indicates dissimilar pairs and $y = 0$ corresponds to similar pairs. L_{sim} is the function which calculates the loss of similar images and is given by $\frac{1}{n}\sum_{i=1}^{n} d_i$. The loss for dissimilar pairs is the average value by which the distances are smaller than the margin m and is defined as $\frac{1}{n}\sum_{i=1}^{n} \max(0, m - d_i)$. The margin m defines how big the Euclidean distance between the feature vectors of negative images should be.

B. Balance Criterion

The first additional criterion added to the loss function is called balance criterion (B). The objective of the criterion is to achieve the same number of 0s and 1s in the generated binary codes and thus to maximize the entropy. Codes which fulfill this condition are called balanced codes. The corresponding loss function is:

$$L_{\mathrm{balance}}(\boldsymbol{X}, y) = y\sum_{i=1}^{n} \left(\mu_b - 0.5\right)^2, \tag{1}$$

The output of the sub-network (one branch of the siamese architecture) which receives the anchor images is $\boldsymbol{X} \in \mathbb{R}^{n\times b}$. The average value of the bits of one binary code $\boldsymbol{x}_i$ is represented by $\mu_b = \frac{1}{b}\sum_{j=1}^{b} x_{ij}$. As the codes become balanced, μ_b will become 0.5 for all codes resulting in minimizing L_{balance}. In equation (1) the label y is also present. The network receives the anchor images twice in each training step. At first it is received along with the positive pair and then with the negative pair. In order to let the network be trained only once with each anchor image with regard to balance, the label y is included. The gradient of L_{balance} with respect to $\boldsymbol{X}$ is:

$$\nabla L_{\mathrm{balance}}(\boldsymbol{X}, y) = \begin{bmatrix} \nabla L_{\mathrm{balance,1}}(\boldsymbol{x}_1, y) \\ \nabla L_{\mathrm{balance,2}}(\boldsymbol{x}_2, y) \\ \vdots \\ \nabla L_{\mathrm{balance,n}}(\boldsymbol{x}_n, y) \end{bmatrix}.$$

$\nabla L_{\mathrm{balance},i}(\boldsymbol{x}_i, y) \in \mathbb{R}^{1\times b}$ is the gradient of the loss function with respect to one binary code $\boldsymbol{x}_i$ and is:

$$\nabla L_{\mathrm{balance},i}(\boldsymbol{x}_i, y) = y\frac{2}{b}\begin{bmatrix}\mu_b, & \mu_b, & \cdots, & \mu_b\end{bmatrix}.$$

The prerequisite for this criterion is, that the codes have an even number of bits. Otherwise balanced codes are not achievable.

C. Orthogonality Criterion

The orthogonality criterion (O) is the second addition to the loss function. This criterion minimizes the mutual information between the bits of the binary codes and thus achieve independence of the bits. In order to achieve this objective, the approach is to generate binary codes that are orthogonal to each other and therefore are linearly independent. [16] proposes a method for balancing the codes in $\{1, -1\}$ domain. We introduce the novel orthogonality criterion which trains the network to generate orthogonal codes in $\{1, 0\}$ domain directly. Let the matrix $\boldsymbol{X}$ contains n binary vectors. The binary vectors are orthogonal if the following condition is met:

$$\boldsymbol{X}\boldsymbol{X}^T - \boldsymbol{C} = \boldsymbol{0},$$

where

$$\boldsymbol{C} = \begin{bmatrix} \frac{b}{2} & \frac{b}{4} & \cdots & \frac{b}{4} \\ \frac{b}{4} & \frac{b}{2} & \cdots & \frac{b}{4} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{b}{4} & \frac{b}{4} & \cdots & \frac{b}{2} \end{bmatrix}.$$

The diagonal entries of the matrix $\boldsymbol{C}$ which are the scalar products of the vectors with themselves must be $\frac{b}{2}$. This is because the scalar product of a binary vector with itself is equal to the sum of ones in the vector and since balanced codes are desired, each code should contain exactly $\frac{b}{2}$ ones. For balanced codes to be orthogonal, the non-diagonal entries have to be $\frac{b}{4}$. For this desired property of the binary codes, a criterion which trains the binary vectors is formulated as given below. The approach is to use the binary vectors generated out of the anchor images $\boldsymbol{X}$ and minimize the loss function:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times b}} L_{\mathrm{orthogonal}}\left(\mathbf{X}, y\right) = y\left\|\mathbf{X}\mathbf{X}^T - \mathbf{C}\right\|^2.$$

The label y is used again to ensure that the network is only trained once with each anchor image.

In order to be able to generate balanced orthogonal binary codes, it is important to select b which is divisible by 4. If the number of bits b is not divisible by 4, $\frac{b}{4}$ would not be an integer and the network will not be trainable. For testing purposes we generate a matrix $\boldsymbol{B}$ containing one binary code for each class. These binary codes are the mean binary codes of all images for each class. 10 binary orthogonal vectors are desired for a database with 10 different classes. A necessary condition is that the vectors are linearly independent. The 10 binary codes in $\boldsymbol{B}$ are linearly independent, if and only if the rank of $\boldsymbol{B}$ is 10. With 10 bits or more it is possible to find 10 balanced binary codes so that the matrix $\boldsymbol{B}$ has a rank of 10. If both

ISBN 978-90-827970-1-5 © EURASIP 2018

requirements are merged together, the resulting number of bits results to $b = 12$ or more. The final loss function is defined as:

$$\text{Loss} = \lambda_1 L_{\text{he}}(\boldsymbol{d}, y) + \lambda_2 L_{\text{balance}}(\boldsymbol{X}, y) + \lambda_3 L_{\text{orthogonal}}(\boldsymbol{X}, y)$$

where λs represent the corresponding regularisers.

D. Selection of margin m

The objective of the orthogonality criterion is to generate binary orthogonal codes. As explained in section II-C, the scalar product between all codes must be $\frac{b}{4}$. This condition can also be formulated differently. When comparing two binary codes, $\frac{b}{2}$ bits should be identical while $\frac{b}{2}$ must be inverted. This means that the Hamming distance between all codes should be equal to $\frac{b}{2}$. The margin m of the HE criterion is therefore set to $\sqrt{\frac{b}{2}}$ to support the orthogonality criterion. We use the square root here because we consider the Euclidean distance during the training. The binary codes are only generated after passing the network output through the sigmoid layer followed by quantization at 0.5.

III. EXPERIMENTAL RESULTS

We have used CIFAR-10 [19] and MNIST [20] datasets for our experiments. For the evaluation of the properties we have trained 20 networks using the image pairs from CIFAR-10 dataset with different criteria and averaged the results. The networks were trained for 150 epochs with a learning rate of 0.01, and batch size of 64. The margin m was selected as explained in II-D. The GPU used for training was GeForce GTX Titan X. Fig. 2 shows the training curves obtained while generating binary codes using all 3 criteria for a bit size of 12. Fig. 2 (a), (b), and (c) shows the training error for dissimilar pairs, similar pairs, and the total training error respectively. Fig. 2 (d) shows how the loss changes as the distance between the similar image features changes. As the training proceeds, the loss decreases and the distance between the similar image features also converges to 0. For dissimilar pairs for the case of $b = 12$, the margin is $\sqrt{\frac{12}{2}} = 2.449$ and as the training proceeds the loss decreases and the distance between the dissimilar pairs moves to the selected margin of 2.449 as shown in Fig. 2 (e). From Fig. 2 (f), we could observe that the distance between dissimilar pairs goes beyond the margin and distance between the similar pairs converge towards zero as the training proceeds.

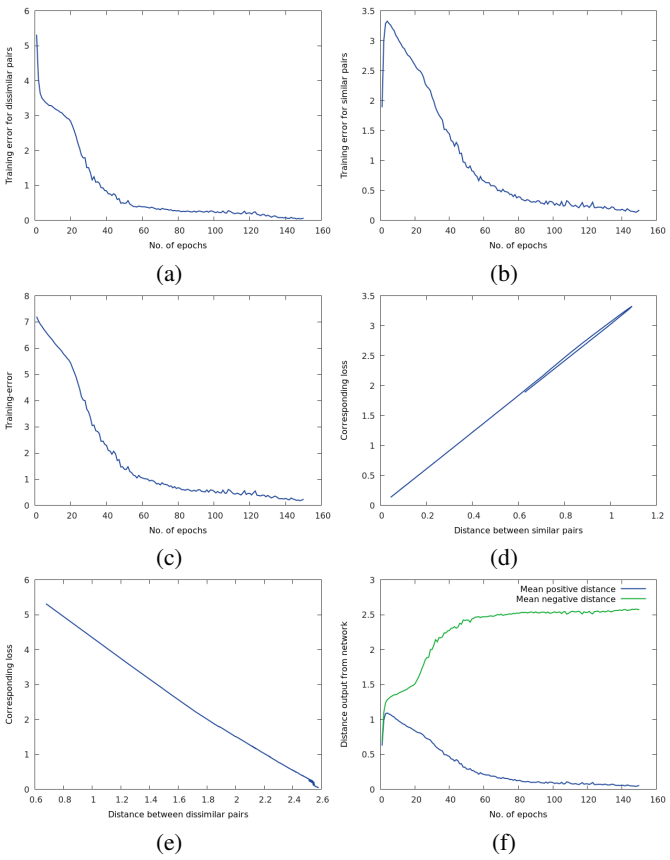

Fig. 2: (a) Training error for dissimilar pairs, (b) training error for similar pairs, (c) total training error, (d) distance between dissimilar pairs vs the corresponding loss, (e) distance between similar pairs vs the corresponding loss, (f) variation of distance between similar pairs (positive) and the dissimilar pairs (negative) as the training proceeds. The bit length used here is $b = 12$.

A. Evaluation of entropy and mutual information

The generated binary codes are to be evaluated with respect to entropy, mutual information, and independence of bits. The entropy of the bit c in position i is defined as $H(c_i)$. In order to calculate the mutual information between 2 bits, the joint entropy of those 2 bits is needed. The joint entropy of the bits in position i and j is calculated as follows:

$$H(c_i, c_j) = -\sum_{k=0}^{1}\sum_{l=0}^{1} p(c_i = k, c_j = l)\log_2(p(c_i = k, c_j = l))$$

The mutual information $I(c_i; c_j)$ is calculated as: $H(c_i) + H(c_j) - H(c_i, c_j)$. The influence of the different criteria on entropy is evaluated first. The entropy of the bits is highest for a network trained with HE+B criterion than one trained with HE (Fig. 3 (a)). The average entropies for HE and HE+B are almost equal. This implies, that the bits carry approximately the same information in both cases. However, it shows that the bits are more balanced for balanced codes. The average entropy value of HE+B+O is in the middle of the other 2 values for 8 bits. For increasing bit sizes

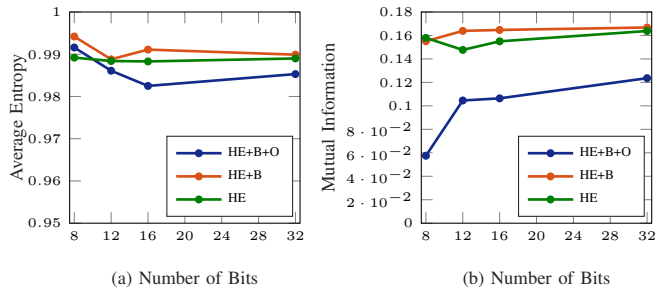

Fig. 3: (a) Variation of entropy, and (b) variation of mutual information with bit sizes.

this still holds true for HE and HE+B, as shown in Fig. 3 (a). The only difference that has to be noted is, that the average entropy for HE+B+O decreases and is on average about 0.005 lower compared to HE and HE+B. Now the effect of different criteria on mutual information is evaluated. As shown in Fig. 3 (b) training the network with the additional orthogonality criterion results in smaller mutual information of the bits. A mutual information close to 0 implies, that 2 bits share no information and are thus totally independent. This is the most desired property the codes should have. Therefore, codes generated with HE+B+O have more bits that are independent. Fig. 3 (b) illustrates how the average mutual information changes,

ISBN 978-90-827970-1-5 © EURASIP 2018

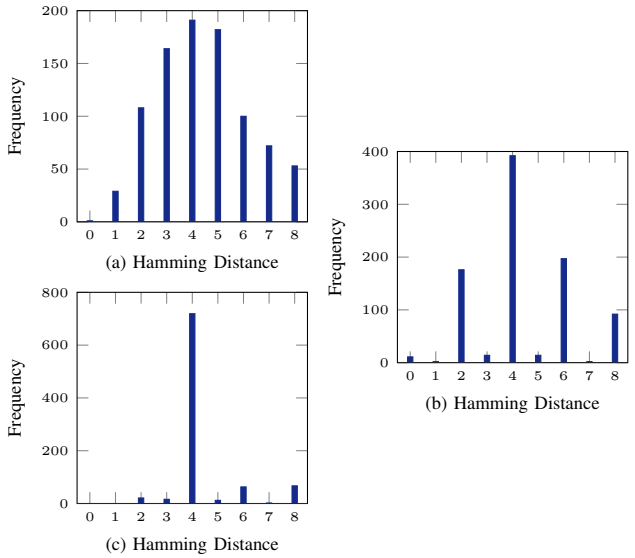

Fig. 4: Histogram of Hamming distances for networks trained with (a) hinge embedding, (b) hinge embedding and balance, (c) all 3 criteria. The network was trained to generate codes for a bit length of 8 (Hamming distance of 4). The best codes were generated when network is trained using all 3 criteria, as seen from the peak in histogram at 4.

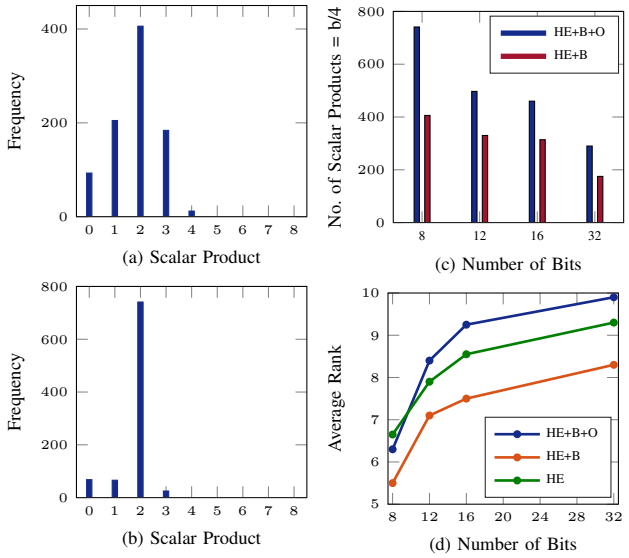

Fig. 5: (a) Scalar product of binary vectors for networks trained with hinge embedding and balance, (b) for networks trained with all 3 criteria for 8 bits. (c) Scalar products which are $\frac{b}{4}$ for different bit sizes. (d) Rank of $\boldsymbol{B}$ for different bit sizes.

when more bits are used to represent the binary codes. The mutual information continues to be lowest with increasing bit sizes when the networks are trained using the orthogonality criteria.

B. Evaluation of Hamming distance

To get an overview of the distances, the binary codes of the 10 classes are compared within each network. For each matrix $\boldsymbol{B}$, $\binom{10}{2} = 45$ Hamming distances are calculated. Fig. 4 (a) shows the 900 Hamming distances occurring for the 20 networks trained with HE for 8 bits. The desired Hamming distance the network is trained for is $\frac{b}{2} = 4$. Although the average Hamming distance is 4.38, many binary codes differ by less than 4 bits. One network actually generates 2 binary codes that are identical, even though they belong to different classes. This is undesired behavior, because the images of those classes can not be distinguished.

For networks trained with HE+B (Fig. 4 (b)) a peak can be seen at a Hamming distance of 4 and less Hamming distances are smaller than 4. There are 11 Hamming distances that are 0, meaning that more binary codes belonging to different classes are identical. Networks trained with HE+B+O (Fig. 4 (c)) provides the best results. It is evident from the peak in the histogram at 4. This is because the orthogonality criterion additionally trains the network to generate codes that differ by exactly 4 bits.

C. Evaluation of orthogonality and linear independence

The main objective of the orthogonality criterion is to reduce the mutual information. Balanced codes are orthogonal to each other if their scalar product is equal to $\frac{b}{4}$. If all codes are orthogonal to each other, all 45 scalar products should be $\frac{b}{4}$. For the 8 bit case, the scalar products should be equal to 2. In Fig. 5 (a) histogram of the scalar products for codes trained with the balance criterion for 8 bits is shown. The number of scalar products that are 2 is higher proving that many binary codes are already orthogonal to each other. However, 54.9% of the scalar products differ from 2. The reason is that the Hamming distance between the codes generated is not always 4 for networks trained with HE+B compared to codes generated with HE+B+O (Fig. 4 (b), (c)). There are a lot more distances that are bigger than 4, because the HE criterion accepts all codes with an Euclidean distance higher than the margin. Fig. 5 (b) shows the histogram of the scalar products for HE+B+O. The histogram shows that many more binary codes are orthogonal to each other. Furthermore 68 out of 900 scalar products are 0, meaning that the corresponding codes are parallel and are linearly dependent.

The scalar product for balanced codes is only $\frac{b}{4}$ if and only if the Hamming distance of the codes is $\frac{b}{2}$. Thus we could conclude that the orthogonality for balanced codes is correlated to the Hamming distance. Therefore, networks trained with the 3 criteria combined generate the best orthogonal codes. Fig. 5 (c) shows the number of scalar products that are $\frac{b}{4}$ for different bit sizes. As readily apparent, the number of binary codes, that are orthogonal to each other, is higher for HE+B+O for all bit combinations. This number is decreasing for higher bit sizes.

Finally, a brief overview about the rank of the matrices $\boldsymbol{B}$ generated with the different criteria is given. The rank specifies how many binary codes in $\boldsymbol{B}$ are linearly independent. Fig. 5 (d) shows the rank for different number of bits. In total, the rank rises for every case. Networks trained with HE+B always have the lowest rank. For 12 bits or higher, networks trained with HE+B+O outperform networks trained with HE. For 16 bits, the average rank is already 9.25. For 32 bits, the average rank for HE+B+O increases further and reaches a value of 9.9, which is really close to the upper limit of 10 for a 10 class problem.

D. Evaluation of retrieval results

TABLE I: MAP values for CIFAR-10 in % compared with other state-of-the-art approaches.

No. of bits	16	24	32	48
ours	90.37	90.44	90.39	90.32
DTSH [21]	91.5	92.3	92.5	92.6
DPSH [22]	76.3	78.1	79.5	80.7
DRSCH [23]	61.5	62.2	62.9	63.1
DSRH [24]	60.8	61.1	61.7	61.8

ISBN 978-90-827970-1-5 © EURASIP 2018

In order to evaluate the retrieval results, we measured the Mean Average Precision (MAP) values. The MAP values are evaluated for CIFAR-10 and MNIST datasets. To make it consistent with the state-of-the-art algorithms, we have measured the MAP values for bit sizes of 16, 24, 32, and 48 bits. The test images were considered as the query images making 1000 query images per class. The query was made on the training dataset. The MAP measurements were done at 1000 meaning first 1000 retrieved images were considered for MAP calculation. The MAP values are summarized in the Table I for CIFAR-10 and Table II for MNIST dataset. The MAP values are for networks trained with all 3 criteria which generates the optimal binary codes. Increase in the number of bits does not bring much improvement in MAP values. For MNIST dataset, binary codes generated using all 3 criteria gives the best MAP for bit size of 16 and does not show much improvement with increasing bit sizes. For CIFAR-10 the MAP value is around 90% slightly lower than the values reported by [21].

TABLE II: MAP values for MNIST in % compared with other state-of-the-art approaches.

No. of bits	16	24	32	48
ours	97.06	97.02	96.49	97.28
DRSCH [23]	96.92	97.37	97.88	97.91
DSRH [24]	96.48	96.69	97.21	97.53

IV. Conclusions

In this paper we have trained a SNN model for generating binary codes with properties such as balance and orthogonality. We introduced the novel orthogonality constraint to generate codes directly in $\{1, 0\}$ domain. The sigmoid layer squashes the neural network output to 1 and 0 and are then finally quantized to generate actual binary codes. Another contribution of our work is the optimum selection of margin m for the hinge embedding criteria. We also evaluated the properties of the binary codes generated by measuring the Hamming distances, entropy, mutual information and rank of the binary codes. To conclude, the networks trained with all 3 criteria together showed promising results especially for smaller number of bits. Properties of binary codes are becoming worse for higher number of bits. The main reason is the general increase in difficulty when dealing with higher bit sizes because more bits have to adapt to the imposed constraints. The second reason is that the λ_3 for the orthogonality criterion has to be reduced for higher bit sizes. Otherwise, the feature vectors fail to assume values close to 0 or 1. Eventually, this could be achieved by initially training the network without the orthogonality criterion and addition of the orthogonality criterion later. After few epochs, the vectors will already be a little bit defined and the orthogonality criterion will only have to do the fine tuning. This could prevent the criterion from completely dominating the initial training phase, while still greatly influencing the training of the network later on.

Another major conclusion is that the scalar product for balanced codes is only $\frac{b}{4}$ only when the Hamming distance of the codes is $\frac{b}{2}$ showing direct correlation between orthogonality and balance. Also from the MAP values and evaluation of properties, it is clear that an increase in bit size is not always required to have distinguishable binary codes with unique properties. Ideally using our orthogonality criterion, for a 10 class problem, 12 bits are sufficient to generate binary codes which are independent. Extension of experiments to bigger datasets and selection of optimum number of bits based on the measured properties of the binary codes are some of the future research directions.

References

[1] 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.

[2] A. Krizhevsky I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.

[3] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors," in IEEE Signal processing magazine, vol. 25, No. 2, pp. 128-131, 2008.

[4] J. Ji, J. Li, S. Yan, B. Zhang and Q. Tian, "Super-bit locality-sensitive hashing," in Advances in Neural Information Processing Systems, 2012, pp. 108-116.

[5] Y. Mu and S. Yan, "Non-Metric Locality-Sensitive Hashing," in AAAI, 2010, pp. 539-544.

[6] Y. Gong, S. Lazebnik, A. Gordo and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, No. 12, pp. 2916–2929, 2013.

[7] W. Liu, J. Wang, R. Ji, Y. Jiang and S. Chang, "Supervised hashing with kernels," in IEEE Computer Vision and Pattern Recognition (CVPR), pp. 2074–2081, 2012.

[8] J. Wang, J. Wang, N. Yu, and S. Li, "Order preserving hashing for approximate nearest neighbor search," in Proceedings of the 21st ACM international conference on Multimedia, pp. 133–142, 2013.

[9] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised Discrete Hashing," in CVPR, vol. 2, No. 3, pp. 5, 2015.

[10] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in AAAI, vol. 1, pp. 2, 2014.

[11] H. Lai, Y. Pan, Y. Liu and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in arXiv preprint arXiv:1504.03410, 2015.

[12] S. Chopra, R. Hadsell, Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification ," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 539–546, 2005.

[13] G. Koch, R. Zemel and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in ICML Deep Learning Workshop, vol. 2, 2015.

[14] V. Erin Liong, J. Lu, G. Wang, P. Moulin, J. Zhou, "Deep hashing for compact binary codes learning," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2475–2483.

[15] K. Lin, J. Lu, C. Chen and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1183–1192.

[16] T. T. Do A. D. Doan, N. Cheung, "Learning to hash with binary deep neural network," European Conference on Computer Vision, in Springer 2016, pp. 219–234.

[17] T. T. Do, A-Z. Doan, N. Cheung, "Discrete hashing with deep neural network," in arXiv preprint arXiv:1508.07148, 2015.

[18] A. Jose, S. Yan, and I. Heisterklaus, "Binary Hashing Using Siamese Neural Networks," in IEEE International Conference in Image Processing, 2017.

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Technical report, University of Toronto, 2009.

[20] Y. LeCun, C. Cortes, C. J. Burges, "MNIST handwritten digit database," in AT&T Labs [Online], http://yann. lecun. com/exdb/mnist, vol. 2, 2010.

[21] X. Wang, Y. Shi and K. M. Kitani, "Deep supervised hashing with triplet labels," in Asian Conference on Computer Vision, pp. 70–84, 2016.

[22] Y. Weiss A. Torralba and R. Fergus, "Spectral hashing," in Advances in neural information processing systems, pp. 1753–1760, 2009.

[23] 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," in IEEE Transactions on Image Processing, vol. 24, No. 12, pp. 4766–4779, 2015.

[24] F. Zhao, Y. Huang, L. Wang and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1556–1564, 2015.

ISBN 978-90-827970-1-5 © EURASIP 2018