Deep Hashing with Hash Center Update for Efficient Image Retrieval

Abin Jose, Daniel Filbert, Christian Rohlfing, and Jens-Rainer Ohm
Institut für Nachrichtentechnik
RWTH Aachen, Germany
jose@ient.rwth-aachen.de

arXiv:2106.06269v1 [eess.IV] 11 Jun 2021

Abstract

In this paper, we propose an approach for learning binary hash codes for image retrieval. Canonical Correlation Analysis (CCA) is used to design two loss functions for training a neural network such that the correlation between the two views to CCA is maximized. The first loss, maximizes the correlation between the hash centers and learned hash codes. The second loss maximizes the correlation between the class labels and classification scores. A novel weighted mean and thresholding based hash center update scheme is proposed for adapting the hash centers in each epoch. The training loss reaches the theoretical lower bound of the proposed loss functions, showing that the correlation coefficients are maximized during training and substantiating the formation of an efficient feature space for image retrieval. The measured mean average precision shows that the proposed approach outperforms other state-of-the-art approaches in both single-labeled and multi-labeled image datasets.

1. Introduction

Due to their excellent feature extraction capabilities, deep neural networks have become state-of-the-art in feature extraction and are the basis of most Content Based Image Retrieval methods [42]. The evaluation of similarity between two images would be performed by computing a predefined distance measure between the feature vectors in the feature space. Besides high retrieval quality, both efficiency and speed are also essential requirements for a good retrieval system. Due to the dramatic and continuous growth of image datasets, evaluation of Euclidean distances for subsequent ranking and nearest neighbor search has become too computationally expensive or even infeasible [35]. In addition, the feature vectors are high-dimensional, which increases the computational complexity exponentially. A solution for this problem would be using compact binary representations. In general, when using neural networks as a feature extractor for image retrieval, two approaches can be considered for obtaining binary codes. In a first approach, an Euclidean feature space can be optimized resulting in feature vectors having well suited properties for retrieval when binarized [19]. Subsequent quantization and binarization of the Euclidean feature space is then mainly to be concerned with minimizing the information loss as well as preserving similarity information in the obtained binary codes. In the second approach, a deep neural network can be trained to directly produce effective binary features [3]. In this case, obtaining binary values also requires the use of an activation function which maps to a binary space in the last layers of the network. We have adopted the second approach in this paper. To overcome the challenges of high computational cost and lack of search speed, Approximate Nearest Neighbour [39] (ANN) search is a common alternative approach which offers more efficiency and sufficiently high accuracy for many practical applications [33]. A widely used form of ANN search is hashing, where a mapping is found to create a lower-dimensional representation of the actual data while preserving the similarity between data points in the new domain accurately. Hashing is in general classified into data-independent approaches and data-dependent approaches. Data independent approaches [22, 17, 29] mainly rely upon random projections to generate the hash functions. Locality-sensitive hashing (LSH), is [6] a popular data-independent approach, which was used in [15] and [9] to solve the ANN problem while avoiding the curse of dimensionality inherently associated with exact nearest neighbor searches for data in their original metric spaces. Recent methods mainly concentrate on, data-dependent approaches which are mainly categorized as unsupervised and supervised methods. We refer readers to [34] for an extensive survey. For learning hash functions, unsupervised approaches [21, 8, 7, 26, 14, 13] use various metrics to supervise the learning. In contrast, the supervised approaches utilize the semantic labels of the training data. In recent years, deep supervised hashing has achieved good retrieval performance [36, 3, 24, 18, 2, 25, 37, 4, 2, 24, 18]. Especially, pair-wise and triplet hashing approaches [3, 41, 36, 25, 37] have shown promising results. However, these

approaches require a lot of time to sample enough pairs or triplets. The loss function depends on the distance calculation between similar and dissimilar pairs and does not utilize the entire feature statistics during training.

1.1. Related work

Quite recently, Canonical Correlation Analysis (CCA) was used such that the correlation between the feature vectors and label vectors was maximized [9]. The learned feature space was then binarized using the popular ITQ [4] approach. Here, statistical properties of the feature space is taken into consideration during the training.

Yuan et al. proposed a new global similarity metric which they called central similarity in [21]. By applying this new metric, hash values of similar data points are encouraged to approach a common center, where as pairs of dissimilar hash codes converge to distinct centers in the Hamming space. There are two systematic approaches proposed in [21] to generate hash centers fulfilling the above condition: One leverages the characteristics of the Hadamard matrix, thus obtaining hash centers with maximal mutual Hamming distance, and the other uses random sampling from a Bernoulli distribution when the bit length is not a power of 2. Generating hash centers from the rows of the Hadamard matrix has several appealing properties: Firstly, it is a binary matrix with elements of value $\{+1, -1\}$ which makes the generation of the hash centers in the Hamming space straightforward. Furthermore, it is a square matrix of size $K \times K$ with $K = 2^n, n \in \mathbb{N}$ being a power of 2 which leads to hash centers with a common amount of bits in the hash codes. In addition, its row vectors are mutually orthogonal. Learning the hash function requires training data to be associated with the generated hash centers in order to reflect semantic information in the Hamming space. There are as many hash centers generated as there are semantic labels in the dataset. However, since each sample can contain one or more categories for multi-labeled data, a majority voting is proposed in order to account for the transitive similarity of data points sharing multiple labels. For training, a central similarity loss is defined which utilizes the binary cross entropy loss to measure the similarity between hash code outputs of the network and its corresponding semantic hash centers. To avoid optimization difficulties implied by the binary valued hash centers, a quantization loss was introduced based on a bi-modal Laplacian prior, as in [22], with additional smoothing. However, a major problem in this approach is that the hash centers are not updated even though the feature space changes during training.

Another interesting work in a similar direction was proposed by Hong et al. in [6]. Here, Linear Discriminant Analysis characteristics are trained directly on the hash codes, thus enforcing the deep network to produce hashes which have a small intra-class variance while also having a high inter-class variance. The proposed method updates the hash centers during deep hashing training. Here, the natural problem arises that hash centers are desired to be binary, such that the CNN features are encouraged to be discrete and are desired to be real valued at the same time, such that gradient descent optimization is feasible [6]. Therefore, a distinct treatment of those hash centers is designed depending on whether performing a forward pass or backward pass step during training is implemented.

2. This paper

Inspired by the idea of updating hash centers during training [6], an alternative method of hash center update based on the weighted mean of the hash values is proposed in our paper, which aims to reflect the movement of the formed clusters in the Hamming space during training. The network was trained using a CCA-based loss formulation such that the correlation between the hash codes and hash centers is maximized along with the correlation of classification scores and class labels. The major contributions of this paper are summarized below:

- First, initial hash centers, around which the hash codes are clustered, are selected as proposed in [21]. A novel weighted mean and thresholding based hash center update scheme is proposed for both single-labeled and multi-labeled images.
- The loss function is formulated using CCA such that the generated hash codes and hash centers have maximum correlation. This loss function is combined with the CCA-based classification loss as proposed in [9] which maximizes the correlation of classification scores with the class labels.
- The theoretical lower bound is determined for both loss functions based on the rank of the two views of CCA, and an optimum regularization factor is chosen to combine the two loss functions.

Experiments were conducted on single-labeled dataset CIFAR-10 [10], as well as multi-labeled datasets MS-COCO [14] and NUS-WIDE [3] and the retrieval performance of the hash codes is evaluated for different bit lengths. The training and test curves, precision-recall curves, and t-SNE [18] plots of the generated feature space were plotted and discussed. Mean average precision was computed and the performance is compared with other supervised hashing approaches such as DPSH [13], DCCH [9], CSQ [21], DSDH [12], DDSH [8], DTSH [19], LDH [6], HashNet [2], DHN [22], DNNH [11], and CNNH [20].

The paper is organized as follows: Section 3 explains the network architecture, and loss function. The hash center

update is discussed in Section 4. Experimental results are discussed in Section 5 and concluding remarks are drawn in Section 6.

3. Proposed approach

An efficient feature space for image retrieval reflects the semantic information contained in its represented images. Deep hashing is used in the proposed Deep Central Similarity Hashing (DCSH) method, which directly learns compact binary hash codes having high correlation with the hash centers representing the image categories. The hash centers are updated during training such that they adapt to the changes in the feature space.

3.1. Network architecture

The network architecture of the proposed approach is given in Fig. 1. The Residual network [5] is used as the basic feature extractor which was pretrained on the ImageNet [16] dataset. Followed by the residual layers, a hashing layer consisting of a fully connected layer and subsequent sigmoid activation function is used to generate hashes corresponding to the given input images. The output dimension of this layer is therefore equal to the required number of bits. An intermediate layer is used subsequently to the hashing layer to generate high output dimensionality from the input hash codes. This intermediate layer is required, since the loss function used in this architecture, L_{DCCF} (see Section 3.3), is a dimensionality reduction method. A higher number of input dimensions exceeding the number of distinct classes in the dataset is required as input to this layer. There are two losses in the proposed approach: 1) L_{hash} correlating the hashing outputs with the semantic hash centers. 2) L_{class} correlating the classification scores with the label information of the dataset. The final training loss, L_{DCSH} then takes both losses into account for the optimization.

3.2. Hash code generation

In proposed DCSH, hashing values are generated by the output of the hashing layer. It contains a sigmoid activation function which produces hashes $\vec{x}_h \in [0,1]^B$ with B elements, and B being the number of required bits for each of the N images in the dataset. After successful training of the network, the values of the hashing outputs are likely to be either close to 0 or 1. An additional thresholding $\tau(\cdot)$ at 0.5 is performed on each element to obtain the desired binary code vectors.

3.3. Loss formulation

The proposed approach uses Canonical Correlation Analysis (CCA) at two output layers of the network to formulate the training loss as shown in Fig. 1. CCA aims to find transformations of two input views which maximally correlates their mapped representations. DCCH [9] uses CCA such that a neural network can be trained to generate non-linear mappings of its input which maximally correlate to a given target. In this paper, this loss formulation is used to optimize the outputs of a neural network from two layers simultaneously. The layers from which the loss is calculated is shown in Fig. 1. For two data views $\mathbf{X}$ and $\mathbf{Y}$, CCA optimizes projections $\vec{a}$ and $\vec{b}$ which maximize the correlation ρ between the projected inputs as:

$$\begin{aligned}\rho(\vec{a}^*, \vec{b}^*) &= \max_{\vec{a},\vec{b}} \operatorname{corr}(\vec{a}^T\mathbf{X}, \vec{b}^T\mathbf{Y}) \\ &= \max_{\vec{a},\vec{b}} \frac{\vec{a}^T\mathbf{\Sigma_{XY}}\vec{b}}{\sqrt{\vec{a}^T\mathbf{\Sigma_{XX}}\vec{a} \quad \vec{b}^T\mathbf{\Sigma_{YY}}\vec{b}}} \\ &\text{s.t. } \vec{a}^T\mathbf{\Sigma_{XX}}\vec{a} = \vec{b}^T\mathbf{\Sigma_{YY}}\vec{b} = 1\ ,\end{aligned} \tag{1}$$

where, $\mathbf{\Sigma_{XX}}, \mathbf{\Sigma_{XY}}$, and $\mathbf{\Sigma_{YY}}$ denotes the covariances, $\operatorname{cov}(\mathbf{X},\mathbf{X}), \operatorname{cov}(\mathbf{X},\mathbf{Y})$, and $\operatorname{cov}(\mathbf{Y},\mathbf{Y})$ respectively. This optimization problem can be solved by using a Singular Value Decomposition, shown by Mardia et al. in [15]. DCCF followed this approach to formulate the loss function as:

$$L_{\mathrm{DCCF}} = -\sum_{i=1}^{k} \sigma_i = -\sum_{i=1}^{k} \rho_i \tag{2}$$

with k largest singular values σ_i of the matrix $\mathbf{K}$ defined by

$$\mathbf{K} := \mathbf{\Sigma_{XX}}^{-1/2}\mathbf{\Sigma_{XY}}\mathbf{\Sigma_{YY}}^{-1/2}\ . \tag{3}$$

The equivalence between the correlation coefficients and the singular values is also proven in [15]. Accordingly, the training loss minimizes the negative sum of correlation coefficients. Since the maximum positive correlation can reach a value of 1, the lower bound of the DCCF loss L_{DCCF} will be $-k$. Furthermore, k can only be as high as the rank of matrix $\mathbf{K}$. For input vectors $\mathbf{X}$ and target data view $\mathbf{Y}$, the upper bound of k is determined by:

$$k_{\max} = \operatorname{rank}(\mathbf{K}) = \min(\operatorname{rank}(\mathbf{X}), \operatorname{rank}(\mathbf{Y})) - 1\ . \tag{4}$$

The subtraction of 1 from either $\operatorname{rank}(\mathbf{X})$ or $\operatorname{rank}(\mathbf{Y})$ is due to the inherent subtraction of the mean for the covariance matrices in $\mathbf{K}$. The loss function of DCSH consists of two CCA evaluations. Both components, the hashing loss and the classification loss, are discussed next.

Hashing loss: First, the DCCF loss is evaluated by using the outputs of the hashing layer $\mathbf{X}_{\mathrm{h}}$ with their corresponding semantic hash centers $\mathbf{Y}_{\mathrm{h}}$. These hash centers act as targets in Hamming space towards which the respective output hashes of the network should converge. The size of the two data views $\mathbf{X}_{\mathrm{h}} \in [0,1]^{M\times B}$ and $\mathbf{Y}_{\mathrm{h}} \in \{0,1\}^{M\times B}$ is determined by the batch size M and the number of bits

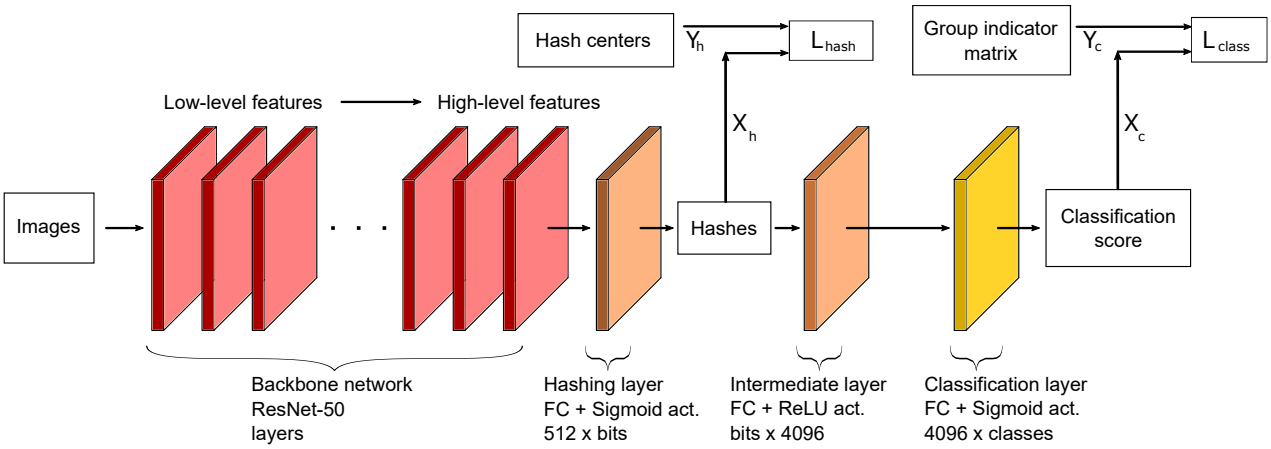


Figure 1: Overview of the proposed Deep Central Similarity Hashing network architecture. ResNet layers according to [5] are used as the backbone network for basic feature extraction. Both hashing and classification layer, consist of a fully connected (FC) layer with subsequent sigmoid activation. The intermediate layer comprises a fully connected layer with subsequent ReLU activation. Bits indicate the bit length of the hash code. Classes indicates the number of categories in the dataset.

B in the binary codes. Therefore, the hashing loss can be formulated as:

$$L_{\text{hash}} = L_{\text{DCCF}}(\mathbf{X}_{\text{h}}, \mathbf{Y}_{\text{h}}) \,. \tag{5}$$

Since each hash center used in the target view $\mathbf{Y}_{\text{h}}$ represents one of the C categories in the underlying dataset, it consists of only at most C distinct rows. Assuming that $M > C$, $\text{rank}(\mathbf{Y}_{\text{h}}) = \min(B, C)$. Therefore, the maximal number of correlation coefficients according to Eq. (4) is:

$$\begin{aligned} k_{\max} &= \min(\min(B, M), \min(B, C)) - 1 \\ &= \min(B, C) - 1 \text{ , for } M > B \text{ and } M > C. \end{aligned} \tag{6}$$

According to Eq. (2), this effectively means that during training the negative sum of $\min(B, C) - 1$ correlation coefficients is minimized. As the correlation maximally can reach a value of 1, the lower bound of the hashing loss goes to $-(\min(B, C) - 1)$.

Classification loss: The second component of the proposed DCSH loss function is a classification loss, which performs a CCA of the classification scores $\mathbf{X}_{\text{c}}$ from the output of the final network layer and a target group indicator matrix $\mathbf{Y}_{\text{c}}$ (see Fig. 1). In this context the first data view $\mathbf{X}_{\text{c}} \in \mathbb{R}^{M \times L}$ is defined as a matrix consisting of M rows each denoting an L-dimensional feature representation $\vec{x}^T \in \mathbb{R}^L$ of a sample of the current batch with M images. As a second data view, a so called group indicator matrix $\mathbf{Y}_{\text{c}} \in \{0, 1\}^{M \times C}$ is used with M rows of label vectors $\vec{y} = [y_1, ..., y_C]$ with C entries (with a value 1 indicating the class associated to the respective image and a value 0 otherwise). This definition can directly be extended for multi-class labels, such that each row of the group indicator matrix $\mathbf{Y}_{\text{c}}$ indicates the associated classes in the corresponding data sample [9]. Note that the classification layer uses a sigmoid activation instead of the usual softmax. This is because the approach aims to be directly extensible to multi-label datasets using the same CCA evaluation. As group indicator matrices are used as the second view, sigmoid activation is an appropriate choice encouraging the network to produce outputs close to 1 for classes contained in the image and 0 otherwise. Here, the size of the two used data views $\mathbf{X}_{\text{c}} \in [0, 1]^{M \times C}$ and $\mathbf{Y}_{\text{c}} \in \{0, 1\}^{M \times C}$ is determined by the batch size M and the number of distinct classes C in the underlying dataset. Given these data views, the classification loss is formulated as:

$$L_{\text{class}} = L_{\text{DCCF}}(\mathbf{X}_{\text{c}}, \mathbf{Y}_{\text{c}}) \,. \tag{7}$$

Assuming a higher batch size than distinct classes $M > C$, the maximal number of correlation coefficients according to Eq. (4) results:

$$\begin{aligned} k_{\max} &= \min(M, C) - 1 \\ &= C - 1 \text{ , for } M > C. \end{aligned} \tag{8}$$

The lower bound of the classification loss then comes to a value of $-(C - 1)$.

Loss combination and normalization: The final training objective of the proposed deep hashing method is designed by a linear combination of the two losses introduced above as:

$$L_{\text{DCSH}} = L_{\text{hash}} + \alpha L_{\text{class}} \,. \tag{9}$$

The regularization factor α balances the contribution of both components for the final optimization objective. By using the theoretical lower bounds of hashing and classification loss as shown in Eq. (6) and Eq. (8), an equal

weighting of their contributions can be computed as:

$$\alpha = \frac{\min(B, C) - 1}{C - 1} \,. \tag{10}$$

Setting α to this value results in the theoretical lower bounds of the hashing and classification loss both becoming equal. However, it has been beneficial in subsequent experiments to choose α in each case of B and C as:

$$\alpha = \frac{B - 1}{C - 1} \,. \tag{11}$$

Therefore, in case of datasets for which $C < B$, the classification loss is emphasized. This occurs for all single-labeled datasets. Experiments on multi-labeled datasets all satisfy $C > B$ and thus, α normalizes the contribution of classification and hashing loss in the final training objective. The theoretical lower bound of the combined DCSH loss can then be determined as follows:

$$\begin{aligned} \min L_{\mathrm{DCSH}} &= \min L_{\mathrm{hash}} + \alpha \min L_{\mathrm{class}} \\ &= -(\min(B, C) - 1) - (B - 1) \,. \end{aligned} \tag{12}$$

4. Hash center update during training

An important part of the training procedure in DCSH is to provide good semantic hash centers, which are updated in each epoch. According to [21], either the Hadamard matrix, in case of required bit length of power of 2, or a Bernoulli distribution $\mathrm{Bern}(0.5)$, for other bit lengths is used to generate the initial hash centers. Both approaches provide hash centers reflecting distinct classes which are sufficiently far apart with respect to the Hamming distance. Each hash center $\vec{h}_c^{(0)}$, $c \in \{1, ..., C\}$ represents one of the C classes in the dataset. The index 0 represents the initial state of the hash centers. These hash centers $\vec{h}_c^{(0)}$, are elements of $\mathbf{Y_h}$, which is one view of CCA loss L_{hash}. Opposed to [21] which keeps hash centers constant during training, our approach additionally performs an update after each training epoch with the goal to better reflect semantic information in Hamming space. Thus, the updated versions are able to better represent the class centers in Hamming space more dynamically. Using this in the context of the DCCF loss results in the target semantic hash centers being able to adapt to the semantic information contained in the actual network outputs. Since a sigmoid activation is used in the hashing layer, the resulting hash values $\vec{x_h}$ will be in the range between $[0, 1]$. These $\vec{x_h}$ are elements of $\mathbf{X_h}$ which is the second view of CCA loss L_{hash}. The following proposed methods for updating hash centers require them to be in the range of $[-1, 1]$. Therefore, we apply a simple mapping, $f(\vec{x_h}) = 2\vec{x_h} - 1$. However, for valid further training the updated hash centers have to be again binary values $\{0, 1\}$. Therefore, a final remapping to the required binary space is performed.

Single-labeled images: After each training epoch, the hash values of each training image are evaluated by a forward pass. By using the information of the class labels, the hashing outputs $\vec{x_h}$ can be grouped in sets $\mathcal{G}_c^{(i)}$, each of which contains the hash values of images associated with class c at epoch i. Now, for each group of hash codes, the mean vector of all hashing values representing the same class is calculated as:

$$\widetilde{\vec{h}}_c^{(i+1)} = \frac{1}{|\mathcal{G}_c^{(i)}|} \sum_{\vec{x_h}^{(i)} \in \mathcal{G}_c^{(i)}} \vec{x_h}^{(i)} \,. \tag{13}$$

A subsequent thresholding of the obtained mean vector $\widetilde{\vec{h}}_c^{(i+1)}$ results in the updated binary hash center $\vec{h}_c^{(i+1)}$, with the thresholding function defined as

$$\tau(\vec{x}) = \begin{cases} 1, \text{ if } x_k \geq 0 \\ 0, \text{ if } x_k < 0 \end{cases} \text{, for each element } x_k \text{ of } \vec{x}. \tag{14}$$

Multi-labeled images: In this case, hash values can be associated with multiple classes at the same time. Therefore, an extension of the hash center update given in eq. (13) is introduced. The hash values are weighted with each update. Hashes are weighted lighter when containing more categories in their labels. This is based on the assumption that with increasing category associations, hash values are less able to represent a single category in Hamming space. Therefore, a weighting factor $w_{\mathrm{n}} = \frac{1}{|l_{\mathrm{n}}|}$ for each of the N hash values is introduced based on the number of classes $|l_{\mathrm{n}}|$ in its corresponding label l_{n}. First, the hashes of the training images are evaluated by a forward pass after weight update of the network in the current epoch. Then, the obtained hash values are grouped in sets $\mathcal{G}_c^{(i)}$, each consisting of those hashes which at least contain the respective class c in their corresponding label. Note, that for multi-labeled data the resulting hash value groups $\mathcal{G}_c^{(i)}$ are not distinct sets as in the previous case of single-labeled data. This is because hash values labeled with multiple categories c at the same time are also contained in each corresponding group $\mathcal{G}_c^{(i)}$. Now, the weights w_{n} are used to compute a weighted mean of the hash values contained in each group $\mathcal{G}_c^{(i)}$ at epoch i:

$$\widetilde{\vec{h}}_c^{(i+1)} = \frac{1}{|\mathcal{G}_c^{(i)}|} \sum_{\vec{x_h}^{(i)} \in \mathcal{G}_c^{(i)}} w_{\mathrm{n}} \, \vec{x_h}^{(i)} \,. \tag{15}$$

By applying a weighted mean, the contribution of each hash value is adapted according to its reflecting semantic information about the class of a group. The steps to perform a hash center update during DCSH training are summarized in Algorithm 4.1.

Algorithm 4.1 DCSH hash center update

Require: Hash centers $\mathcal{H}^{(i)} = \{\vec{h}_c^{(i)}\}, c = 1, ..., C$ with C classes, and output hashes $\{\vec{x_h}^{(i)}\} \in [0; 1]^n$ with $n = 1, ..., N$ from N training images at epoch i.

1: For each class c, group all hash outputs associated with c in their label l_{n}:
$\mathcal{G}_c^{(i)} = \{\vec{x_h}^{(i)} : c \text{ in } l_{\mathrm{n}}\}$, for $c = 1, ..., C$.

2: Calculate weights for each output hash based on number of classes in its label $|l_{\mathrm{n}}|$:
$w_{\mathrm{n}} = \frac{1}{|l_{\mathrm{n}}|}$.

3: Calculate weighted mean of grouped hashing values:
$\tilde{\vec{h}}_c^{(i+1)} = \frac{1}{|\mathcal{G}_c^{(i)}|} \sum_{\vec{x_h}^{(i)} \in \mathcal{G}_c^{(i)}} w_{\mathrm{n}} \, \vec{x_h}^{(i)}$, for $c = 1, ..., C$.

4: Create updated binary hash centers by thresholding:
$\vec{h}_c^{(i+1)} = \mathrm{sign}(\tilde{\vec{h}}_c^{(i+1)})$.

5: **return** Updated hash centers $\mathcal{H}^{(i+1)}$ for epoch $i + 1$.

5. Experimental results

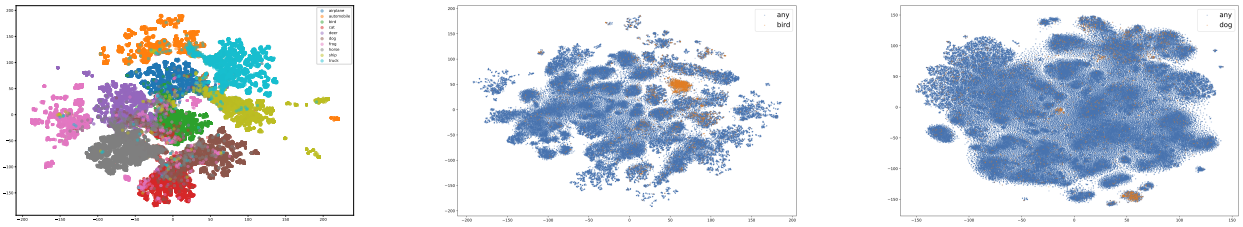
In this paper, the single-labeled dataset CIFAR-10 [10], and multi-labeled datasets MS-COCO [14] and NUS-WIDE [3] were used for evaluating the final retrieval performance. For each dataset, the training and test curves and precision-recall (P-R) curves were plotted. Furthermore, the retrieval performance was measured by calculating mean average precision (MAP) [8]. The neural network architecture used is ResNet-50 [5]. We used a batchsize of 200 in our experiments, with an initial learning rate and learning rate decay of 0.0003 and 0.7 every 10^{th} epoch for CIFAR-10 and 0.0008 and 0.1 for multi-labeled datasets which was determined by grid search. The optimizer used is Stochastic gradient descent. The regularizer α was chosen as shown in Eq. (12).

CIFAR-10 [10] is a labeled subset of tiny images [17] with 10 classes. Each class is equally represented by 6000 images making a total of $60,000$ images. In order to better reflect the requirements of an image retrieval application, the following setup is used in accordance to the experimental setup in DDSH [8]. From the total $60,000$ images, 1000 images are used as query images by sampling 100 random images from each of the ten classes of the dataset. The remaining $59,000$ images constitute the gallery set against which retrievals are performed. Furthermore, 5000 training images are randomly sampled from the gallery set such that each class is again equally represented with 500 images. Thus, the training set is a labeled subset of the gallery set. The experiments were run for 25 epochs. The training and test loss during optimization for binary codes of 32 bits is depicted in Fig. 3 (a). (The loss curves and P-R curves for other bits are given in the supplementary material.) The theoretical lower bound of -40 as mentioned in Eq. (12) is reached during training. The distribution of the resulting binary hashes in Hamming space is visualized in Fig. 2 (a) by using t-SNE for the gallery images. Binary features associated with the same class are located close to each other while having good separability to binary features of other categories. Furthermore, semantically related categories like 'automobile', 'airplane' or 'truck' are located next to each other in Hamming space, while categories like 'dog' or 'cat' are located on the other side. The MAP@5000 was measured and the results are outlined in Table 1. It can be seen that the proposed DCSH method clearly outperforms existing state-of-the-art approaches. Furthermore, P-R curves for 32 bits was plotted in Fig. 3 (d). P-R curves were generated by retrieving all relevant images in gallery whereas MAP values in Table 1 denotes MAP@5000 in which 5000 returned images was considered. Here, the precision is high for a large range of recall values. The precision only starts to decrease for high values of recall. Therefore, the majority of relevant images with respect to a given query image are likely to be returned within the top retrievals.

Table 1: MAP@5000 for CIFAR-10.

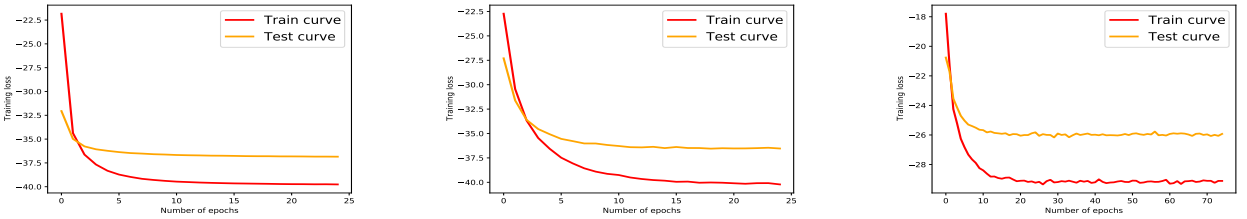
Method	12 bits	24 bits	32 bits	48 bits
DCSH (Ours)	**0.863**	**0.898**	**0.902**	**0.911**
DCCH [9]	0.794	0.830	0.841	0.851
DDSH [8]	0.769	0.829	0.815	0.819
DSDH [12]	0.740	0.786	0.801	0.820
LDH [6]	-	-	0.784	0.792
DTSH [19]	0.710	0.750	0.765	0.774
DPSH [13]	0.713	0.727	0.744	0.757

MS-COCO [14] is a multi-labeled dataset where each image is associated with several of 80 distinct object categories. The used data split for image retrieval evaluation is based on the setup described in HashNet [2]. Combining training and validation images and subsequent discard of all images without category information results in a total number of $122,218$ images. $5,000$ query images are randomly sampled from these obtained images, leaving the remaining $117,218$ images as the gallery set. In addition, $10,000$ images are randomly sampled from the gallery set for training. The model was trained for 25 epochs. The training and test loss, values during training for 32 bits is given in Fig. 3 (b). Theoretically, the lower bound of the loss function is $-2(B - 1)$ with B number of bits, as discussed in Eq. (12). However, it can be seen that this lower bound is not reached. This is because a multi-labeled dataset cannot be fully class-wise clustered as each image may belong to several categories. Therefore, the correlation coefficients of the underlying loss are not able to reach the maximum.

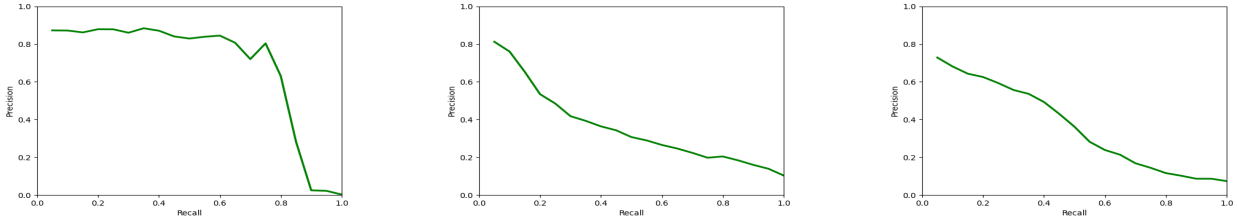


(a) CIFAR-10 gallery features (b) MS-COCO bird category in orange (c) NUS-WIDE dog category in orange

Figure 2: T-SNE of 64 dimensional features for a) gallery images of CIFAR-10, b) MS-COCO highlighting bird category, and c) NUS-WIDE highlighting dog category.



(a) CIFAR-10 loss curve (b) MS-COCO loss curve (c) NUS-WIDE loss curve



(d) CIFAR-10 P-R curve (e) MS-COCO P-R curve (f) NUS-WIDE P-R curve

Figure 3: First row indicates the training and test loss for all the three datasets for 32 bits. The training loss reaches the lower bound of -40 for CIFAR-10. For multi-labeled, the lower bound of -62 could not be reached as the correlation coefficients will never reach the maximum value. Second row indicates the P-R curves for all the three datasets for 32 bits.

The training reaches a lower bound indicating formation of an optimum feature space. Since MS-COCO is a multi-labeled dataset, it is not feasible to depict all class associations for the data points at once by using different colors for each categories. Instead, single categories have been selected to be highlighted against the remaining categories of the dataset. Clusters of binary hashes are formed in the feature space. There are cases where these clusters represent a unique category, as clearly shown in the example of 'bird' depicted in Fig. 2 (b). The retrieval performance of DCSH for MS-COCO in terms of MAP is given in Table 2. Results are compared to state-of-the-art approaches. It can be seen that CSQ is having same MAP for 64 bits and in all other cases, the proposed method outperforms other retrieval frameworks. CSQ did not provide the MAP for the case of 48 bits in [21]. The P-R curves for 32 bits are shown in Fig. 3 (e). The top 3 retrieval results for a query image with labels 'surfboard' and 'person' are shown in Fig. 4.

Table 2: MAP@5000 for MS-COCO.

Method	16 bits	32 bits	48 bits	64 bits
DCSH (Ours)	**0.805**	**0.847**	**0.859**	**0.861**
CSQ [21]	0.796	0.838	-	**0.861**
DCCH [9]	0.659	0.729	0.731	0.739
HashNet [2]	0.687	0.718	0.730	0.736
DHN [22]	0.677	0.701	0.695	0.694
DNNH [11]	0.593	0.603	0.604	0.610
CNNH [20]	0.564	0.574	0.571	0.567



(a) Query image 1 (b) Retrieval result 1 (c) Retrieval result 2 (d) Retrieval result 3

Figure 4: Query image and top 3 retrievals on MS-COCO, all contain category labels 'surfboard' and 'person'.



(a) Query image 1 (b) Retrieval result 1 (c) Retrieval result 2 (d) Retrieval result 3

Figure 5: Query image and top 3 retrievals on NUS-WIDE, all contain category label 'person'.

NUS-WIDE was introduced by Chua et al. in [3] containing $269,648$ web images in total which are associated with $5,018$ unique tags from Flickr. It is a multi-labeled dataset in which each image is associated with one or more of 81 concepts. The experimental setup for NUS-WIDE in DDSH [8] is used here. Only the images belonging to the 10 most frequent concepts are used. This results in a total number of $186,577$ images, from which $1,867$ query images are randomly sampled [8]. The remaining $184,710$ images constitute the gallery set, and $5,000$ images are randomly sampled from it for training [8]. The network was trained for 75 epochs. Fig. 3 (c) shows the training and test loss during training for 32 bits. The theoretical lower-bound in this case is $-2(B-1)$ as well, and cannot be reached as explained in the case of MS-COCO. In Fig. 2 (c), orange color indicates the 'dog' category. It can be seen that there is clearly a dominant area in the feature space. Therefore, a nearest neighbor search would result in retrieved images very likely being associated with the similar semantics. MAP has been measured for the proposed approach and the numbers were compared to state-of-the-art approaches. The results are outlined in Table 3. DCSH is mainly compared to CSQ [21] and DCCH [9]. The proposed DCSH method outperforms other approaches. Note that CSQ did not list the MAP for 12 and 24 bits in [21]. The MAP for 16 bits is given in [21] and is 0.810, for which DCSH gave a value of 0.828. For further evaluation, the precision-recall curves were plotted. For DCSH for 32 bits P-R curve is given in Fig. 3 (f). The precision drops as recall increases as many wrong retrievals occur till we retrieve all relevant images. An example query image and the top 3 retrieval results for category 'person' are shown in Fig. 5.

Table 3: MAP@5000 for NUS-WIDE.

Method	12 bits	24 bits	32 bits	48 bits
DCSH (Ours)	**0.823**	**0.833**	**0.841**	**0.857**
DPSH [13]	0.794	0.822	0.838	0.851
DCCH [9]	0.782	0.814	0.825	0.834
CSQ [21]	-	-	0.825	0.832
DSDH [12]	0.776	0.808	0.820	0.829
DDSH [8]	0.791	0.815	0.821	0.827
DTSH [19]	0.773	0.808	0.812	0.814
LDH [6]	0.769	0.789	0.787	0.803

6. Conclusions

In this paper, we have proposed an approach for learning efficient hash codes for image retrieval. The neural network is trained using a loss function in such a way that the correlation between the hash codes and hash centers is maximized. Canonical Correlation Analysis (CCA) is utilized in the loss formulation, and hash codes and hash centers are chosen as the two views of CCA. The network is also trained using the classification loss as well, which maximizes the correlation between category labels and classification scores. The hash centers are then dynamically updated so that the hash centers adapt to the changes in feature space. The experimental results on both single-labeled and multi-labeled datasets substantiates the generation of an optimized feature space with minimum intra-class scatter and maximum inter-class scatter. This is in fact possible due to the inherent equivalence between Linear Discrimi-

nant Analysis [7] and CCA as proven in [1]. As a future research direction, more effective representations of individual categories could be explored for multi-labeled datasets, which in turn could lead to better retrieval results.

References

[1] Maurice S Bartlett. Further aspects of the theory of multiple regression. In *Proceedings of the Cambridge Philosophical Society*, volume 34, pages 33–40, 1938.

[2] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, pages 5608–5617, 2017.

[3] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–9, 2009.

[4] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 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, 2012.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Weixiang Hong, Yu-Ting Chang, Haifang Qin, Wei-Chih Hung, Yi-Hsuan Tsai, and Ming-Hsuan Yang. Image hashing via linear discriminant learning. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2531–2539, 2020.

[7] Alan Julian Izenman. Linear discriminant analysis. In *Modern multivariate statistical techniques*, pages 237–280. Springer, 2013.

[8] Qing-Yuan Jiang, Xue Cui, and Wu-Jun Li. Deep discrete supervised hashing. *IEEE Transactions on Image Processing*, 27(12):5996–6009, 2018.

[9] Abin Jose, Erik Stefan Ottlik, Christian Rohlfing, and Jens-Rainer Ohm. Optimized feature space learning for generating efficient binary codes for image retrieval. *arXiv preprint arXiv:2001.11400*, 2020.

[10] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[11] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3270–3278, 2015.

[12] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. *arXiv preprint arXiv:1705.10999*, 2017.

[13] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*, 2015.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[15] Kantilal Varichand Mardia. Multivariate analysis. Technical report, 1979.

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[17] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

[18] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[19] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. In *Asian conference on computer vision*, pages 70–84. Springer, 2016.

[20] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

[21] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3083–3092, 2020.

[22] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.