Cross-Modal Retrieval Using Deep De-correlated Subspace Ranking Hashing

Kevin Joslyn
University of Central Florida
Orlando, Florida
KevinJoslyn@knights.ucf.edu

Kai Li
University of Central Florida
Orlando, Florida
kaili@cs.ucf.edu

Kien A. Hua
University of Central Florida
Orlando, Florida
kienhua@cs.ucf.edu

ABSTRACT

Cross-modal hashing has become a popular research topic in recent years due to the efficiency of storing and retrieving high-dimensional multimodal data represented by compact binary codes. While most cross-modal hash functions use binary space partitioning functions (e.g. the *sign* function), our method uses ranking-based hashing, which is based on numerically stable and scale-invariant rank correlation measures. In this paper, we propose a novel deep learning architecture called Deep De-correlated Subspace Ranking Hashing (DDSRH) that uses feature-ranking methods to determine the hash codes for the image and text modalities in a common hamming space. Specifically, DDSRH learns a set of de-correlated nonlinear subspaces on which to project the original features, so that the hash code can be determined by the relative ordering of projected feature values in a given optimized subspace. The network relies upon a pre-trained deep feature learning network for each modality, and a hashing network responsible for optimizing the hash codes based on the known similarity of the training image-text pairs. Our proposed method includes both architectural and mathematical techniques designed specifically for ranking-based hashing in order to achieve de-correlation between the bits, bit balancing, and quantization. Finally, through extensive experimental studies on two widely-used multimodal datasets, we show that the combination of these techniques can achieve state-of the-art performance on several benchmarks.

CCS CONCEPTS

• **Information systems** → **Multimedia and multimodal retrieval**; *Learning to rank*; *Top-k retrieval in databases*; • **Computing methodologies** → Neural networks;

KEYWORDS

Multimodal retrieval; cross-modal hashing; image and text retrieval

ACM Reference Format:
Kevin Joslyn, Kai Li, and Kien A. Hua. 2018. Cross-Modal Retrieval Using Deep De-correlated Subspace Ranking Hashing. In *ICMR '18: 2018 International Conference on Multimedia Retrieval, June 11–14, 2018, Yokohama, Japan.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3206025.3206066

ICMR '18, June 11–14, 2018, Yokohama, Japan

ACM ISBN 978-1-4503-5046-4/18/06…$15.00
https://doi.org/10.1145/3206025.3206066

1 INTRODUCTION

The prevalence of digital and social media in modern society has created a world that is dominated by multimedia. For billions of people around the globe, images, stories, social posts and even videos are just a query and a click away. A relatively new but already popular research area inspired by the fresh abundance of multimedia is cross-modal hashing for multimodal retrieval. The problem is defined as follows: given a query from a certain modality (such as image or text), the goal is to retrieve relevant results from a database of another modality. Typical examples of cross-modal retrieval are using text to retrieve relevant images or vice versa (e.g. Google image search).

The K-nearest neighbors (kNN) problem is a classic problem that involves retrieving the k database items that are most similar to the query. The main issue with kNN is the high computational costs that can be attributed to the curse of dimensionality, especially when dealing with multimedia such as high-dimensional images and videos. In order to avoid making costly comparisons among examples in the original feature space, hashing for similarity search has become a popular dimensionality reduction technique. Through hashing, examples can be compared much more efficiently in a discrete (rather than continuous) hamming space of a much lower dimensionality.

Locality Sensitive Hashing (LSH) [10] introduced the concept of hashing for similarity search. Initially, methods in the LSH family relied on choosing random hyperplanes to separate the data points, where each hyperplane accounts for one hash bit and the hash bit is determined according to which side of the hyperplane the data point belongs. LSH is an example of a *data-independent* hashing method, since the hash function depends on randomization rather than the actual data. *Data-dependent* hashing methods are now the more common variety- these methods use either supervised or unsupervised learning algorithms to learn meaningful hash functions that are tailored to the data.

The problem of cross-modal retrieval is especially complex because it requires the comparison of two objects that exist in entirely different feature spaces. Thus, cross-modal hashing solutions must either determine an effective mapping from one modality to the other modalities, or a mapping from each modality to a common hamming space. In this work, we focus on the latter approach. Most cross-modal hashing methods generate hash codes by discretizing a continuous output space, most commonly by using the *sign* function to create a binary code. Instead, this work is motivated by the ideas introduced in [16], called *Linear Subspace Ranking Hashing* (LSRH). The idea is to learn a unique K-dimensional linear subspace for each hash "bit", such that we may generate a K-ary hash code. To determine each hash bit, the original data point is projected onto

the corresponding K-dimensional subspace, and the hash bit is set equal to the index of the dimension with the maximum projection.

Although LSRH has achieved competitive performance on a number of cross-modal retrieval benchmarks, there are several limitations that need to be addressed. First, the use of strictly linear subspaces prevents the optimality that may be achieved by exploiting the nonlinearities of deep neural networks. Second, there is no explicit way for LSRH to make sure that multiple hash bits are uncorrelated, and it has to rely on boosting to implicitly reduce the code redundancy. Third, the hash codes generated by LSRH are not balanced, which can cause an underutilization of the available information capacity of an L- bit hash code. Finally, LSRH has no unified method for feature representation across modalities or datasets. The use of hand-crafted features (1) undermines the generalizability of the model, and (2) neglects the potential of supervised deep learning for feature representation that has proven to be powerful in many recent hierarchical deep learning methods [3–5, 12, 24].

In this paper, we introduce a novel deep neural network architecture for cross-modal hashing called Deep De-correlated Subspace Ranking Hashing (DDSRH) that addresses the above limitations of LSRH. First, by making use of deep neural networks, DDSRH is able to learn ranking subspaces that are more optimal due to their nonlinearity. We address correlation between the bits (i.e. bit redundancy) by abandoning the typical fully-connected model in favor of fully-connected sub-layers, with one sub-layer for each $K-$ary hash bit. Next, we counter the effects of unbalanced bit distributions by introducing a regularization term designed to encourage bit balance. A second regularization term is also added to improve the quantization performance of the model, which actually competes with the bit balance term in an effort to achieve an optimal balance of the terms. Finally, we create a unified feature representation by employing pre-trained image and text feature extractors to precede the hashing network and feed it with meaningful features that have benefited from supervised deep learning.

In this paper, we describe each of the aforementioned components in detail and evaluate the effectiveness of each technique by using each one in isolation and in combination with the others (see Section 4.3, Ablation Study.) We show that the overall model not only greatly outperforms LSRH, but furthermore achieves state-of-the-art results on two widely used datasets for cross-modal retrieval.

In summary, the major contributions of this paper are as follows:

- We propose a novel deep neural network architecture that is the first to use de-correlated nonlinear subspace ranking hashing for cross-modal retrieval.
- We introduce novel de-correlation, bit balancing, and quantization techniques that are specifically designed for hashing using subspace ranking, and evaluate the performance gains obtained by using each technique.
- We demonstrate through several experiments that DDSRH obtains state-of-the-art performance on two widely used datasets for cross-modal retrieval.

The remainder of the paper is organized in the following manner. Section 2 discusses related work, including both linear and nonlinear/deep cross-modal hashing methods. Section 3 describes our method in detail, including the architecture of the neural network and loss function formulation. Section 4 discusses the experiments and results, and Section 5 concludes the work.

2 RELATED WORK

The earliest work on cross-modal hashing was Cross-Modal Similarity Sensitive Hashing (CMSSH) [2], which uses boosting to sequentially learn linear hash functions for each modality. Other early cross-modal hashing methods include Cross-View Hashing (CVH) [15] and Co-Regularized Hashing (CRH) [28]. Since these pioneering works, cross-modal hashing research has split into several directions, summarized here.

Some methods, including Collective Matrix Factorization Hashing (CMFH) [8] and Latent Semantic Sparse Hashing (LSSH) [29] use matrix factorization methods to learn hash functions that bridge the modalities. Other methods were designed with scalability considerations at the forefront, including Inter-media hashing (IMH) [22], Semantic Correlation Maximization (SCM) [27], and Semantic Topic Multimodal Hashing (STMH) [23]. Quantized Correlation Hashing (QCH) [25] and Composite Correlation Quantization (CCQ) [19] are examples of methods that use quantization techniques for cross-modal hashing.

An emerging category of cross-modal hashing algorithms uses a two-stage learning framework that separates binary code learning from hash function learning. In the first stage, the binary codes for each training instance are discretely optimized by minimizing a loss function based on the similarity of the instances. In the second stage, the hash functions are learned for each modality by treating each of the optimized bits as a classification problem. Semantics-Preserving Hashing (SePH) [18] is a well-known approach that first learns the hash codes by minimizing the KL-divergence between the cross-modal similarity distribution and the hash code distribution. SePH then uses logistic regression to learn the hash functions for each modality. Label Preserving Multimedia Hashing (LPMH) [17] introduces a general loss function and optional bit balancing term to be used for the hash code learning, and it makes use of boosted decision trees for the hash function learning stage. Semantic Neighbor Graph Hashing (SNGH) [13] introduces a semantic neighbor graph to guide the hash code learning by introducing a fine-grained similarity metric based on the neighborhood structure of the graph and the semantic similarity of the instances.

Another increasingly popular direction for cross-modal hashing algorithms involves deep learning. Masci et al. [20] were the first to use parallel neural networks for cross-modal hashing. The general structure, involving one neural network for the text modality and a parallel network for the image modality, has been imitated and modified by more recent works, including the proposed DDSRH. Of these similar works, Deep Cross-Modal Hashing [12] was the first method to use deep neural networks to learn features and hash codes simultaneously. During each training iteration, similar image-text pairs are given the same hash code while just the image or the text feature parameters are learned. In the third step of each iteration, the image and text feature parameters are held constant while the hash codes are learned. Correlation Hashing Network (CHN) [3] was the first network to consider cosine max-margin loss for deep hashing methods, as well as quantization max-margin loss, which acts as a regularizer. Note that rather than learning

feature and hash codes simultaneously, CHN pre-trains an image feature extractor and a text feature extractor, both of which become components of the final model. Our proposed method follows this same approach. Collective Deep Quantization (CDQ) [4] is the successor to CHN. The main difference is the use of a collective quantization "codebook" that is shared across modalities, which was seen to improve the quantizability of the deep representations and quality of the resulting hash codes.

One drawback of all of the aforementioned methods is that they are all dependent on exact feature values, which makes them arguably more susceptible to noise and variations. By contrast, our method belongs to a relatively new class of hash functions based on ranking operations, which are closely related to rank correlation measures. These techniques in particular are known to be robust against noise and variations [16, 26]. Even so, ranking-based hashing for cross-modal retrieval remains a very understudied area. Linear Subspace Ranking Hashing (LSRH) [16] was the first method to use ranking-based hash functions for the cross-modal retrieval problem. Inspired by Winner-Take-All (WTA) Hash [26] and Min-wise Hash [1], LSRH generates K-ary hash bits that correspond to the maximum feature dimension for each feature subspace. Rather than using randomized subspaces in the original feature space like WTA and MinHash, LSRH learns optimal subspaces in a new feature space by using boosting for each linear hash function. To date, LSRH remains the only work of its kind to use ranking-based methods for cross-modal hashing, and it serves as the basis for this work.

In summary, the proposed DDSRH bridges the gap between the deep learning and ranking-based categories of cross-modal hashing algorithms, incorporating the core advantages of both. The following section explains the proposed method in detail.

3 DEEP DE-CORRELATED SUBSPACE RANKING HASHING

3.1 Problem Definition

Let $\mathcal{D}_{\mathcal{X}} = \mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N_{\mathcal{X}}}$ and $\mathcal{D}_{\mathcal{Y}} = \mathbf{Y} = \{\mathbf{y}_j\}_{j=1}^{N_{\mathcal{Y}}}$ be datasets of the image and text modalities, respectively, where $\mathbf{x}_i \in \mathcal{D}_{\mathcal{X}}$ is a $d_{\mathcal{X}}$-dimensional image and $\mathbf{y}_j \in \mathcal{D}_{\mathcal{Y}}$ is a $d_{\mathcal{Y}}$-dimensional text (set of tags). We also define $\mathbf{S} = \{s_{ij}\} \in \{0,1\}^{N_{\mathcal{X}} \times N_{\mathcal{Y}}}$ to be a set of similarity labels, where $s_{ij} = 1$ if $\mathbf{x}_i$ is similar to $\mathbf{y}_j$ (by belonging to the same concept), and 0 otherwise. The goal is to learn two sets of hash functions $H_* = \{h_*^{(l)}\}_{l=1}^{L}$, one for each modality, to transform the input images and texts into a common K-ary Hamming space, such that similar image-text pairs are hashed to codes that are close together in Hamming space while dissimilar image-text pairs are hashed to codes that are far apart. In this paper, * is a placeholder for the modality when we need not refer to one modality specifically (thus it can refer to modality $\mathcal{X}$ or $\mathcal{Y}$), and likewise $\mathbf{Z}_*$ refers to either dataset $\mathbf{X}$ or $\mathbf{Y}$. Finally, $\mathbf{W}_*$ represents the learned parameters for the given modality.

In the following sections, K refers to the number of subspace dimensions we seek to optimize, resulting in a K-ary hash code. L refers to the number of K-ary digits in a hash code, while $L_B = L \times \lceil \log_2 K \rceil$ is the number of binary bits used to represent the code. N refers to the number of examples in question (for example, training instances) The softmax function is denoted by $\sigma(x)$.

3.2 Ranking-Based Hashing

We consider a different category of ranking-based hash functions other than the well-studied sign or thresholding-based functions. Formally, the ranking-based hash function is defined as follows:

$$h(\mathbf{z}_*; \mathbf{W}_*) = \underset{\mathbf{h}}{\operatorname{argmax}}\, \mathbf{h}^T \phi(\mathbf{z}_*; \mathbf{W}_*) \quad \text{s.t. } \mathbf{h} \in \{0,1\}^K, \mathbf{1}^T\mathbf{h} = 1 \tag{1}$$

where $\mathbf{z}_* \in \mathcal{D}_*$ is the input sample (text or image) and $\phi(\mathbf{z}_*)$ is an embedding of the input. Note that the 1-of-K constraint ensures that there is only a single 1 in the output hash code corresponding to the maximum entry of the embedded data point, while the remaining entries are 0s. Therefore, one instance of the above hash function encodes a K-ary hash code corresponding to the maximum index of data embeddings. To obtain a length-L hash code, one needs to apply the hash function to L different embeddings.

Hash functions based on ranking operations are closely related to rank correlation measures which are known to be robust against noise and variations [16, 26]. In fact, as long as the implicit ordering of embedded features remains the same, the hash code will not change.

Note that the hash functions used in LSRH and WTA are special cases of the above hash function. LSRH defines each embedding function as $\phi(\mathbf{z}_*; \mathbf{W}_*) = \mathbf{W}_*^T \mathbf{z}_*$, where $\mathbf{W}_* \in \mathbb{R}^{K \times d_*}$ defines a K-dimensional linear subspace used to rank the projected features. On the other hand, WTA's embedding function can be realized by using the same embedding function but setting the columns of $\mathbf{W}_*$ equal to columns randomly selected from a $d_* \times d_*$ identity matrix [16].

3.3 DDSRH

From (1), note that both LSRH and WTA use linear embedding functions for ϕ. DDSRH instead uses a nonlinear embedding function ϕ that can be used to reveal more discriminative nonlinear ranking structures that cannot be exposed in linear subspaces. We accomplish this by designing a deep neural network architecture to learn the nonlinear ranking subspaces. Such a design allows us to take advantage of both rank correlation measures and deep learning structures to learn the most discriminative ranking-based hash functions.

DDSRH also makes use of novel de-correlation, bit balancing and quantization approaches that are designed specifically for the K-ary encoding scheme and the subspace ranking problem. By comparison, LSRH does not consider bit balancing or quantization loss, and it has to rely upon boosting for bit de-correlation rather than explicit bit de-correlation which is made possible with DDSRH. In the following subsection, we introduce the objective function and its components, which are responsible for ensuring cross-modal similarity, bit balancing, and quantization. We leave bit de-correlation to section 3.8 because it is an architectural modification rather than a term in the objective function.

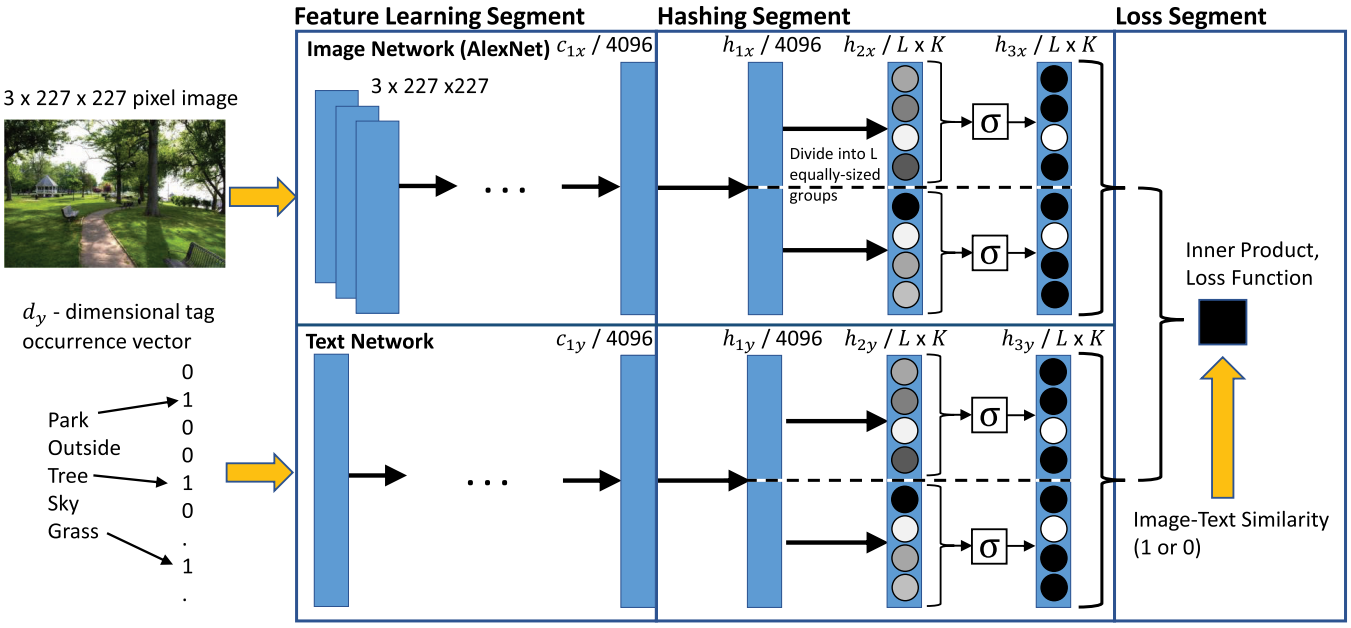


Figure 1: Full Network structure for deep cross-modal hashing. The network consists of three segments: the feature learning segment, the hashing segment, and the loss segment. Inputs to the network are given by the wide arrows, while solid black arrows indicate fully-connected layers or groups. Layers are given a label (e.g. $h_{1\mathcal{X}}$) and a dimension. The softmax function is given by σ, and is applied to groups of neurons of size K. Neuron values are depicted in grayscale, with black indicating low values and white indicating high values (relative to the group). Assuming that the first neuron in each group is at index 0, the corresponding hash code for both the image and the text would be the quaternary code 21.

3.4 Objective Function

The objective function has three components: a cross-modal similarity loss term, a bit balancing term, and a quantization loss term.

3.4.1 Cross-Modal Similarity. Each cross-modal training pair is given a binary similarity label s_{ij} that defines whether the pair is similar or not similar. Incorporating the ranking hash function h from (1), we define the loss term for a single training pair as

$$l(\mathbf{x}_i, \mathbf{y}_j) = \left(\frac{1}{L}\mathbf{b}_{i\mathcal{X}}^T \mathbf{b}_{j\mathcal{Y}} - s_{ij}\right)^2, \tag{2}$$

where $\mathbf{b}_{i\mathcal{X}}, \mathbf{b}_{j\mathcal{Y}} \in \{0,1\}^{LK} = [h^1(\cdot)^T\ h^2(\cdot)^T\ ..\ h^L(\cdot)^T]^T$ are the vectorized hash codes for image $\mathbf{x}_i$ and text $\mathbf{y}_j$ respectively, and $h^l(\cdot)$ denotes $h^l(\mathbf{x}_i; \mathbf{W}_{\mathcal{X}})$ for images and $h^l(\mathbf{y}_j; \mathbf{W}_{\mathcal{Y}})$ for texts respectively. Note that although the length of the hash code is represented as $L \times K$ bits only for mathematical convenience, the actual binary code length is only $L \times \log_2 K$.

We can interpret (2) as follows. First, it is easy to see that the result of the subtraction operation is a scalar between $[-1, 1]$, due to the constraints placed on $\mathbf{h}$ in (1). If the image-text pair are similar, we would like the result of $\mathbf{b}_{i\mathcal{X}}^T \mathbf{b}_{j\mathcal{Y}}$ to be very close to L so that the loss goes to 0. For this to occur, the positions of the '1' values in $\mathbf{b}_{i\mathcal{X}}$ and $\mathbf{b}_{j\mathcal{Y}}$ should be the same, indicating an agreement on the indices of the maximum embedded feature dimensions across modalities. Otherwise, if the image-text pair are dissimilar, we would like the result of $\mathbf{b}_{i\mathcal{X}}^T \mathbf{b}_{j\mathcal{Y}}$ to be very close to 0, indicating a difference in the positions of the '1' values in $\mathbf{b}_{i\mathcal{X}}$ and $\mathbf{b}_{j\mathcal{Y}}$. Ultimately, this criterion accomplishes the goal of pushing the hash codes for similar image-text pairs close together, and pulling the hash codes of dissimilar image-text pairs far apart.

The overall similarity loss term $\mathcal{S}$ is the aggregate similarity loss for all $s_{ij} \in \mathbf{S}$ and is given as follows:

$$\mathcal{S}(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}}) = \left\| \frac{1}{L}\mathbf{B}_{\mathcal{X}}{}^T \mathbf{B}_{\mathcal{Y}} - \mathbf{S} \right\|_F^2, \tag{3}$$

where $\mathbf{S} \in \{0,1\}^{N \times N}$ is the similarity matrix with s_{ij} defined as in 3.1, and $\mathbf{B}_* \in \{0,1\}^{LK \times N}$ is the matrix of vectorized hash codes for all samples. Note that $\mathcal{S}$ is a function of $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$ due to the use of the hash function $h(\mathbf{z}_*; \mathbf{W}_*)$.

3.4.2 Bit Balancer. In addition to the similarity term $\mathcal{S}$, we will introduce a bit balancing term $\mathcal{B}$ to balance the distribution of the K-ary digits in the hash code and thus prevent underutilization of the information capacity of the code. Since most hashing algorithms deal with binary hash codes, they need only to balance the occurrence of 1's and 0's (or 1's and -1's) in the code. Thus one can achieve balance simply by minimizing $|\mathbf{B} \cdot \mathbf{1}|$, where $\mathbf{B} \in \{-1, 1\}^{L \times N}$ is the matrix of binary codes and $\mathbf{1}$ is is a vector of 1's of size N.

Binary codes are actually a special case for our algorithm, which must attempt to balance the occurrence of all $\{0, 1, .., K-1\}$ in a K-ary hash code. Then, for each bit position $l \in \{1, 2, ..L\}$ in the hash code, we would like each $d \in \{0, 1, .., K-1\}$ to occur roughly N/K times across the N instances. We can achieve this by minimizing (4):

$$\mathcal{B}(\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}) = \left\| \mathbf{B}_\mathcal{X} \cdot \mathbf{1} - \frac{N}{K} \cdot \mathbf{1}' \right\|_F^2 + \left\| \mathbf{B}_\mathcal{Y} \cdot \mathbf{1} - \frac{N}{K} \cdot \mathbf{1}' \right\|_F^2, \quad (4)$$

where $\mathbf{B}_* \in \{0,1\}^{LK \times N}$, $\mathbf{1}$ is a vector of 1's of size N, and $\mathbf{1}'$ is a vector of 1's of size LK. For further explanation, let $\mathbf{C} = \mathbf{B}_* \cdot \mathbf{1} \in \mathbb{R}^{LK \times 1} \geq 0$. Then $\mathbf{C}_i$ counts the number of occurrences of the K-ary digit $d = i \mod K$ in bit position $l = i/L$ across all N instances.

3.4.3 Quantization. For binary hash functions, quantization loss terms aim to achieve a maximum margin between instances classified on either side of the threshold value. Given the classical hash function $h(\mathbf{X}) = \text{sign}(\mathbf{W}_\mathcal{X}^T \mathbf{X})$, a good quantizer would encourage $\mathbf{W}_\mathcal{X}^T \mathbf{X}$ to be far from 0 to minimize the risk of similar instances being on either side of the threshold value and obtaining different hash codes. Of course, our algorithm does not use a threshold value to quantize the bits; rather, each hash code bit is assigned by using the argmax function over K values. Accordingly, our quantization loss term aims to maximize the gap between the maximum value and the other $K-1$ values. Note that although the implementation of our quantization loss term differs from the classical binary thresholding case, the motivation is the same: we wish to minimize the risk of similar instances obtaining different bit values by a narrow margin, while also minimizing the risk of dissimilar instances obtaining the same bit values by a narrow margin.

Because the formal equation is best explained after the network structure has been introduced, in (5) we simply refer to the quantization loss term as $\mathcal{Q}(\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y})$. It is formally defined in (7).

3.4.4 Overall Objective Function. The overall objective is to learn parameters $\mathbf{W}_\mathcal{X}$ and $\mathbf{W}_\mathcal{Y}$ that minimize

$$\min \ \mathcal{S}(\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}) + \alpha \mathcal{B}(\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}) + \lambda \mathcal{Q}(\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}), \quad (5)$$

where $\mathcal{S}$ is the similarity loss term defined in (3), $\mathcal{B}$ is the bit balancing term defined in (4), $\mathcal{Q}$ is the quantization loss term to be defined in (7), and α and λ are hyperparameters of the algorithm. The following section describes how to optimize this equation.

3.5 Optimization

Unfortunately, due to the discontinuous and non-convex nature of the argmax term in (1), the objective function in (5) is difficult to optimize. Thus, we will reformulate (1) by using the softmax function to create a continuous probabilistic approximation to the ranking hash function:

$$h(\mathbf{z}; \mathbf{W}) \approx \sigma(\phi(\mathbf{z}; \mathbf{W})), \quad (6)$$

where σ represents the softmax function, and $*$'s are omitted to remove notational clutter. Now h still outputs a vector of size K, but the output values are continuous values between $[0, 1]$ that sum to 1. Importantly, the smoothness of the softmax approximation can be tuned such that its output converges to the binary indicator vector $\mathbf{h}$ in (1).

In order to achieve a good set of nonlinear embedding functions $\{\phi_*^{(l)}\}_{l=1}^L$ for each modality, we propose the following hierarchical deep neural network which is the first to use de-correlated nonlinear subspace ranking hashing for cross-modal retrieval. We optimize the embeddings outputted by the neural network by combining the proposed similarity and bit balancing terms with novel adaptations of de-correlation and quantization measures for subspace ranking hashing, described in the following sections.

3.6 Network Architecture

Our deep neural network consists of two independently pre-trained feature learning networks, one for each modality, each followed by a hashing segment and connected by a cross-modal loss segment. A depiction of the full network during the cross-modal training stage is given in Fig. 1. In the figure, image-text pairs are shown being fed into the network along with a similarity label (inputs are given by wide arrows). A detailed explanation of each segment of the network is given in the following sections.

3.7 Feature Learning Segment

The feature learning segment is constructed by pre-training two independent classifier networks, one for each data modality. The structure of the image classifier is an AlexNet [14] convolutional neural network (CNN) with 5 convolutional layers and 3 fully-connected layers. The text classifier is a deep neural network with three hidden layers, each of size 2048. A rectifying linear unit (ReLU) and dropout are employed after each hidden layer.

Each classifier is trained on the appropriate modality of the dataset using binary cross entropy loss. After the classifiers have been trained, the features of the second-to-last fully-connected layer (denoted in Fig. 1 as layer c_{1*}) are useful as input to the hashing segment of the network. Thus, we effectively remove the output layer of the classifiers (the third fully-connected layer), resulting in the feature learning segment shown in Fig. 1. Layer c_{1*} is then connected to the hashing segment, described in the next section.

3.8 Hashing Segment

The hashing segment's purpose is to determine the hash value of the input image or text. It does this by learning L distinct K-dimensional subspaces, where L is the length of the K-ary hash code to be determined. Thus the output of the hashing segment is effectively the projection of the input onto each of the L latent subspaces.

Fig. 1 shows that the layers in the hashing segment can each be divided into L equally-sized groups, where each group corresponds to one subspace being learned and thus one hash bit. (In Fig. 1 we only show two such groups to conserve space). Thus layer h_{1*} is divided into L groups of size $4096/L$, while layers h_{2*} and h_{3*} are each divided into L groups of size K. Note that for each of the L groups of neurons in layer h_{2*}, we use the softmax function to transform the values such that for each group in layer h_{3*}, the sum of the outputs in that group is 1. To obtain the hash value of the input image/text, each group of neurons in layer h_{3*} reports the index of the neuron with the maximum output (shown by the white neurons in Fig. 1.) Assuming that the first neuron in each group is at index 0, the corresponding hash code for both the image and the text in Fig. 1 would be the quaternary code 21 (in practice the code would be longer for a network with more than two groups of neurons.)

3.8.1 De-correlation. It is important for each hash bit to contain information that is de-correlated, or independent from, the other hash bits in order to utilize the full information capacity of the hash code. This requirement motivates an important design characteristic of our deep network. Looking again at Fig. 1, observe that while layers c_{1*} and h_{1*} are fully-connected, only corresponding *groups* of neurons are fully- connected between layers h_{1*} and h_{2*}. Since each group is responsible for one hash bit, the lack of interconnections between the groups should greatly reduce information redundancy and correlation between the hash bits.

3.8.2 Quantization Loss Term. In section 3.4.3, we explained in non-mathematical terms the motivation and reasoning behind our quantization approach. In order to design a quantization loss term to suit our method, notice that each group of K outputs in layer h_{3*} sums to exactly 1. In the ideal case, we would like one output to be exactly 1, and the rest to be 0. To see how this relates to sign-thresholding in the binary quantization case, note that we can essentially reduce our approach to the binary case when we set $K = 2$. Then, the quantized value is determined by thresholding either output at 0.5 rather than 0.

Regardless of the subspace dimension K, we can encourage one output in h_{3*} to be close to 1 and the rest to be close to 0 by minimizing the following equation:

$$Q(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}}) = -\left\|\mathbf{B}_{\mathcal{X}} - 0.5\right\|_1 - \left\|\mathbf{B}_{\mathcal{Y}} - 0.5\right\|_1 \tag{7}$$

where $\mathbf{B}_* = \sigma(\mathbf{W}_*^T \mathbf{Z}_*) \in [0,1]^{LK \times N}$ is the output of the final hashing layer h_{3*}. Since Q subtracts the distance from 0.5 for all elements of $\mathbf{B}_{\mathcal{X}}$ and $\mathbf{B}_{\mathcal{Y}}$, the quantization term is minimized when all of the outputs of h_{3*} are exactly 0 or 1.

3.9 Training the Network

Training consists of two stages: classifier pre-training, and cross-modal training. For the image classifier, we use a replica of AlexNet that has been pre-trained by The Berkeley Vision and Learning Center [1]on the ImageNet [21] dataset using Caffe [11], and fine-tune the weights for each of the experimental datasets. The text classifier weights are randomly initialized and trained only on each of the experimental datasets.

The proposed DDSRH model is implemented using Torch [7]. In the cross-modal training stage, the weights learned during the classifier pre-training stage are kept relatively constant by setting a low learning rate of 10^{-6} to all of the weights in the feature learning segment. Learning rates for the weights in the hashing segment begin at .05. After 10 epochs, this learning rate for these weights is decreased to .01 and after 50 epochs it is decreased to .005.

The cross-modal training stage is accomplished by feeding image-text pairs into the network, along with a similarity indicator (1 or 0). Note that the same validation and test sets that are used for the classifier pre-training phase are also used for the cross-modal training phase. In this work, we use stochastic gradient descent (SGD) to optimize the weights for our model. All models are trained using a momentum coefficient of 0.9 and without weight decay. We use a batch size of 200 training pairs during training, where 50 pairs are similar pairs and 150 pairs are dissimilar pairs. Using a validation set, we have selected α to be .015 and λ to be .25.

[1]Pretrained model is publicly available for download at https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet

4 EXPERIMENTS

4.1 Datasets

Brief descriptions of the datasets are given in the following subsections. Please note that the terms *label* (or *class label*) and *concept* are interchangeable. Also, note that the term *image-text couplet* specifically refers to an image from the database and its associated textual tags, while the term *training pair* instead refers to a image and a set of tags that are not necessarily associated.

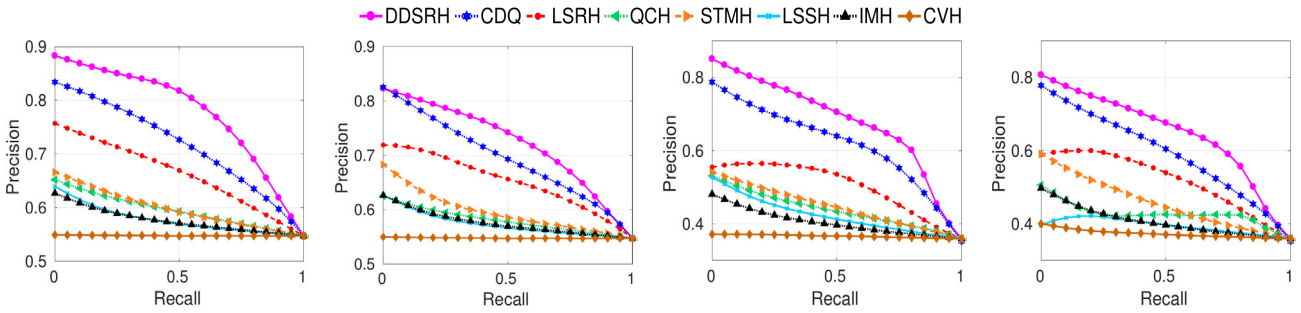
4.1.1 MIRFLICKR. The MIRFLICKR-25000 dataset [9] consists of 25,000 image-text couplets, each of which is annotated with one or more of 24 semantic labels. Textual tags that occur less than 20 times are removed, and then couplets without tags are removed, resulting in 18,159 image-text couplets. Each set of tags is represented by a 1,075-dimensional bag-of-words vector, representing the 1,075 most common tags across the dataset. Following the experimental setup in CHN and CDQ [3, 4], we randomly select 1,000 image-text couplets for the query set, 1,000 couplets for the validation set, and 4,000 couplets for the training set.

4.1.2 NUS-WIDE. The NUS-WIDE dataset [6] consists of 269,648 image-text couplets, each of which is annotated with one or more of 81 concepts. Following the experimental setup in CHN and CDQ [3, 4], we use the 195,834 image-text couplets that belong to the 21 largest concepts. Tags are represented by a 1,000 bag-of-words vector that corresponds to the 1,000 most frequent tags. Again, following CHN and CDQ, we randomly select 500 couplets per concept for the training set, 100 couplets per concept for the query set, and 50 couplets per concept for the validation set. Couplets are only selected once to avoid duplicates.

4.1.3 Training Pairs. For the single-modality classifier pre-training phase, all images and texts except those that exist in the query set or validation set are used for training. During the cross-modal training phase, a *training pair* consists of any one image and any one text from the training set. *Dissimilar* training pairs are chosen at random by choosing pairs of images and texts that share no class labels. Rather than choosing the *similar* training pairs at random, we employ a similarity threshold function that is based on the following notion: similar training pairs whose image and text component share multiple class labels are more likely to be useful for training the network than similar pairs that share only one or two of many class labels. Thus we choose the *similar* training pairs by using the following similarity threshold function:

$$\tau(\mathbf{x}_i, \mathbf{y}_j) = \frac{C_i^T C_j}{0.5 * (sum(C_i) + sum(C_j))}, \tag{8}$$

where C_i and C_j are the binary class label vectors for image $\mathbf{x}_i$ and text $\mathbf{y}_j$ respectively, and $sum(C_i)$ is the number of classes to which example $\mathbf{x}_i$ belongs. Higher values of τ then indicate that $\mathbf{x}_i$ and $\mathbf{y}_j$ share a large fraction of the classes to which they belong. Since such pairs with high τ values are likely to be more useful for training the network, we only use similar training pairs that satisfy $\tau > 0.5$. We also use all dissimilar pairs where, trivially, $\tau = 0$.



(a) MIRFLICKR, Text to Image (b) MIRFLICKR, Image to Text (c) NUS-WIDE, Text to Image (d) NUS-WIDE, Image to Text

Figure 2: Precision-recall curves using a 32-bit hash code. DDSRH outperforms the other baselines in all cases.

Table 1: Mean average precision at top 50 (mAP@50) results for our method versus baseline methods. The best results are shown in bold. The bit lengths given refer to the number of binary bits in the hash code L_B. All methods were tested using the same data splits and results reported are the average over five runs.

	MIRFLICKR						NUS-WIDE					
	Text query Image			Image query Text			Text query Image			Image query Text		
Method	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
QCH	0.7055	0.7015	0.6911	0.6651	0.6682	0.6655	0.6407	0.5993	0.5683	0.5383	0.6014	0.6171
STMH	0.6718	0.7055	0.7341	0.6844	0.7158	0.7370	0.5942	0.6081	0.6499	0.6274	0.6689	0.7200
LSSH	0.6934	0.7106	0.7160	0.6455	0.6679	0.6851	0.6034	0.6235	0.6468	0.5651	0.5716	0.5609
IMH	0.6878	0.6875	0.6823	0.6862	0.6883	0.6884	0.5344	0.5647	0.5731	0.5624	0.5990	0.6101
CVH	0.5957	0.6149	0.6395	0.5797	0.5834	0.5834	0.4304	0.4090	0.3972	0.5023	0.4886	0.4794
LSRH	0.7709	0.7825	0.7959	0.7323	0.7393	0.7465	0.5658	0.5799	0.5621	0.6437	0.6381	0.6383
CDQ	0.8429	0.8517	0.8567	**0.8414**	**0.8495**	**0.8542**	0.7900	0.8090	0.8102	0.7921	0.8115	0.8136
DDSRH	**0.8894**	**0.9020**	**0.9037**	0.8297	0.8431	0.8505	**0.8657**	**0.8887**	**0.8991**	**0.8159**	**0.8412**	**0.8463**

4.2 Comparison with Baselines

The retrieval task is divided into two categories: image-query-text and text-query-image. The first uses images as queries to retrieve the top k relevant texts from the database, and the second performs the opposite task. Here, we compare the performance of our model against other works by using mean average precision obtained at the top 50 results (mAP@50), as well as precision-recall curves.

Table 1 shows the mAP@50 results for our method versus CDQ [4], a state-of-the art deep nonlinear hashing method, and several linear hashing methods: QCH [25], STMH [23], LSSH [29], IMH [22], CVH [15], and LSRH [16]. For each experiment, we vary the number of bits from 16 to 64. All methods were trained and tested using our data splits. For the linear hashing methods, the input is the deep features outputted from our feature learning segment. For CDQ, we selected the most optimal parameters by using the same validation set used to tune our own model. It is important to note that every experiment in this paper is run five times so that each reported result is the average across the five runs.

DDSRH outperforms the compared methods by approximately 5% or more on both datasets for the text-query-image task. For the image-query-text task, DDSRH still shows competitive performance. In fact, on the NUS-WIDE dataset, DDSRH exhibits the highest image-query-text performance by a margin of approximately 4%. This shows that DDSRH can perform equally well on both large datasets such as NUS-WIDE and small datasets such as MIRFLICKR.

We believe that the overall lower performance seen on the image-query-text task is due to the semantic gap between images and texts in the datasets. In both datasets, texts belonging to the same category are more similar to each other than images belonging to the same category. As noted in [16, 29], since texts are better able to describe the semantic concept than images, higher mean average precision can be expected for text queries.

Fig. 2 shows the precision-recall curves for DDSRH compared to each baseline. It is evident that DDSRH obtains the best performance. Note that in each case, the precision for DDSRH remains relatively high until the recall reaches approximately 0.75, indicating that DDSRH is capable of returning more relevant results even for larger queries when many items are retrieved.

4.3 Ablation Study

To evaluate the effectiveness of the de-correlation, quantization, and bit balance components of our method, we ran several experiments in which each component was either utilized or not utilized. To eliminate the bit balance or quantization components, we simply set α or λ in (5) to 0 respectively. To eliminate the de-correlation component, we modify the neural network architecture such that

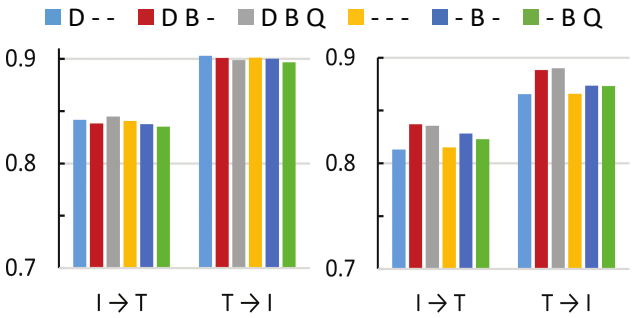


Figure 3: mAP @ 50 observed by varying which features are turned "on" among de-correlation (D), bit balancing (B), and quantization (Q). In the legend, a capital letter denotes that the corresponding feature is turned "on". Image-query-text (I → T) and text-query-image (T → I) results are shown for both MIRFLICKR (a) and NUS-WIDE (b).

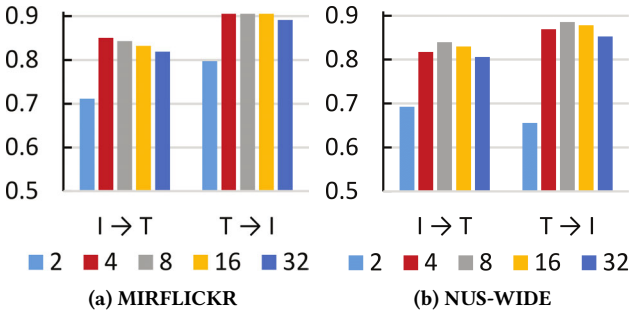


Figure 4: mAP @ 50 observed by altering the subspace dimension K, while keeping the bit length of the hash code fixed at $L_B = 60$. Image-query-text (I → T) and text-query-image (T → I) results are shown for both MIRFLICKR (a) and NUS-WIDE (b).

h_{1*} and h_{2*} in Fig. 1 are fully connected. The experiments were run using a hash code length of 32 bits.

The mAP@50 results of this experiment are shown in Fig. 3. The legend indicates which features have been turned "on", where D = de-correlation, B = bit balance, and Q = quantization. Observing the results for NUS-WIDE, we find that the de-correlation component can be very powerful when coupled with bit balancing. While adding de-correlation alone or bit balancing alone only improves the performance by a small amount, adding both together improves the mAP by almost 3% for both retrieval tasks. The results for MIRFLICKR do not indicate the same improvements, which we speculate is due to the fact that it is a much smaller dataset and thus the model can still benefit from a fully-connected architecture with fewer regularization terms.

Finally, we observe that the addition of the quantization term does not consistently improve the mAP results. We believe that this is because the similarity loss term from (3) can only be truly minimized when the outputs of h_{3*} are exactly 0 or 1, such that no quantization loss would occur. Thus, even though the similarity loss term is designed to enforce similarity learning, it appears to be sufficient enough for quantization as well. Note that the results shown in Fig. 2 and Table 1 were gathered using de-correlation, bit balancing, and quantization (see Section 3.9 for the hyperparameter values that determine the weight of the bit balancing and quantization components.) However, as can be seen from Fig. 3, similar results could be achieved by setting λ in (5) to 0 and effectively removing the quantization component from the loss function.

4.4 Effect of Subspace Dimension

This experiment studies the effect of varying the different subspace dimension K. The experiment is designed such that for each value of K, the amount of information contained in the hash code is held constant by using the same number of binary bits, $L_B = 60$. To accomplish this, we test $K = 2^1, 2^2, .., 2^5$ and set the number of K-ary digits L as $L = 60/\log_2 K$. ($L_B = 60$ is chosen since 60 is a common multiple of 1 to 5.) Fig. 4 shows the effect of changing the subspace dimension K. It is evident that K values between 4 and 16 perform well for both MIRFLICKR and NUS-WIDE. This is similar to the findings in LSRH, which found the optimal subspace dimension to be 4 across several datasets. However, it is advisable to use cross-validation to choose the optimal subspace dimension for the given dataset.

5 CONCLUSION

In this paper, we have proposed a novel ranking-based cross-modal hashing method called Deep De-correlated Subspace Ranking Hashing, the first of its kind to exploit de-correlated nonlinear ranking-based hashing for cross-modal retrieval. The nonlinear, de-correlated subspaces learned by the deep neural network were proven to be much more effective at preserving the cross-modal similarity than the linear subspaces proposed in prior works. Additionally, we have proposed adaptations of de-correlation, bit balancing, and quantization for ranking-based hashing; and we have demonstrated that de-correlation and bit balancing in particular can significantly improve the performance of the model. Comparing DDSRH to several modern baselines, we have shown that DDSRH significantly outperforms current state-of-the-art methods on the text-query-image task, while achieving competitive or superior performance on the image-query-text task.

ACKNOWLEDGMENTS

This work is partially supported by Crystal Photonics, Inc. under research ID 1063271. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of Crystal Photonics, Inc.

REFERENCES

[1] Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. 2000. Min-wise independent permutations. *J. Comput. System Sci.* 60, 3 (2000), 630–659.

[2] Michael M Bronstein, Alexander M Bronstein, Fabrice Michel, and Nikos Paragios. 2010. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 3594–3601.

[3] Yue Cao, Mingsheng Long, and Jianmin Wang. 2016. Correlation hashing network for efficient cross-modal retrieval. *arXiv preprint arXiv:1602.06697* (2016).

[4] Yue Cao, Mingsheng Long, Jianmin Wang, and Shichen Liu. 2017. Collective Deep Quantization for Efficient Cross-Modal Retrieval.. In *AAAI*. 3974–3980.

[5] Yue Cao, Mingsheng Long, Jianmin Wang, Qiang Yang, and S Yu Philip. 2016. Deep Visual-Semantic Hashing for Cross-Modal Retrieval.. In *KDD*. 1445–1454.

[6] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. NUS-WIDE: a real-world web image database from National University of Singapore. In *Proceedings of the ACM international conference on image and video retrieval*. ACM, 48.

[7] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

[8] Guiguang Ding, Yuchen Guo, and Jile Zhou. 2014. Collective matrix factorization hashing for multimodal data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2075–2082.

[9] Mark J Huiskes and Michael S Lew. 2008. The MIR flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*. ACM, 39–43.

[10] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 604–613.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 675–678.

[12] Qing-Yuan Jiang and Wu-Jun Li. 2017. Deep Cross-Modal Hashing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[13] Lu Jin, Kai Li, Hao Hu, Guo-Jun Qi, and Jinhui Tang. 2018. Semantic Neighbor Graph Hashing for Multimodal Retrieval. *IEEE Transactions on Image Processing* 27, 3 (2018), 1405–1417.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[15] Shaishav Kumar and Raghavendra Udupa. 2011. Learning hash functions for cross-view similarity search. In *IJCAI proceedings-international joint conference on artificial intelligence*, Vol. 22. 1360.

[16] Kai Li, Guojun Qi, Jun Ye, and Kien Hua. 2016. Linear subspace ranking hashing for cross-modal retrieval. *IEEE transactions on pattern analysis and machine intelligence* (2016).

[17] Kai Li, Guo-Jun Qi, and Kien A Hua. 2017. Learning label preserving binary codes for multimedia retrieval: A general approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14, 1 (2017), 2.

[18] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. 2015. Semantics-preserving hashing for cross-view retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3864–3872.

[19] Mingsheng Long, Yue Cao, Jianmin Wang, and Philip S Yu. 2016. Composite correlation quantization for efficient multimodal retrieval. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 579–588.

[20] Jonathan Masci, Michael M Bronstein, Alexander M Bronstein, and Jürgen Schmidhuber. 2014. Multimodal similarity-preserving hashing. *IEEE transactions on pattern analysis and machine intelligence* 36, 4 (2014), 824–830.

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

[22] Jingkuan Song, Yang Yang, Yi Yang, Zi Huang, and Heng Tao Shen. 2013. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 785–796.

[23] Di Wang, Xinbo Gao, Xiumei Wang, and Lihuo He. 2015. Semantic Topic Multimodal Hashing for Cross-Media Retrieval.. In *IJCAI*. 3890–3896.

[24] Wei Wang, Beng Chin Ooi, Xiaoyan Yang, Dongxiang Zhang, and Yueting Zhuang. 2014. Effective multi-modal retrieval based on stacked auto-encoders. *Proceedings of the VLDB Endowment* 7, 8 (2014), 649–660.

[25] Botong Wu, Qiang Yang, Wei-Shi Zheng, Yizhou Wang, and Jingdong Wang. 2015. Quantized Correlation Hashing for Fast Cross-Modal Search.. In *IJCAI*. 3946–3952.

[26] Jay Yagnik, Dennis Strelow, David A Ross, and Ruei-sung Lin. 2011. The power of comparative reasoning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2431–2438.

[27] Dongqing Zhang and Wu-Jun Li. 2014. Large-Scale Supervised Multimodal Hashing with Semantic Correlation Maximization.. In *AAAI*, Vol. 1. 7.

[28] Yi Zhen and Dit-Yan Yeung. 2012. Co-regularized hashing for multimodal data. In *Advances in neural information processing systems*. 1376–1384.

[29] Jile Zhou, Guiguang Ding, and Yuchen Guo. 2014. Latent semantic sparse hashing for cross-modal similarity search. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 415–424.