# Deep Learning to Hash with Multiple Representations

Yoonseop Kang[1], Saehoon Kim[1], Seungjin Choi[1,2,3]
[1] *Department of Computer Science and Engineering,*
[2] *Division of IT Convergence Engineering,*
[3] *Department of Creative IT Excellence Engineering,*
*Pohang University of Science and Technology, Pohang 790-784, Korea*
*Email: {e0en,kshkawa,seungjin}@postech.ac.kr*

***Abstract*—Hashing seeks an embedding of high-dimensional objects into a similarity-preserving low-dimensional Hamming space such that similar objects are indexed by binary codes with small Hamming distances. A variety of hashing methods have been developed, but most of them resort to a single view (representation) of data. However, objects are often described by multiple representations. For instance, images are described by a few different visual descriptors (such as SIFT, GIST, and HOG), so it is desirable to incorporate multiple representations into hashing, leading to *multi-view hashing*. In this paper we present a deep network for multi-view hashing, referred to as *deep multi-view hashing*, where each layer of hidden nodes is composed of *view-specific* and *shared* hidden nodes, in order to learn individual and shared hidden spaces from multiple views of data. Numerical experiments on image datasets demonstrate the useful behavior of our deep multi-view hashing (DMVH), compared to recently-proposed multi-modal deep network as well as existing shallow models of hashing.**

***Keywords*-deep learning; harmonium; hashing; multi-view learning; restricted Boltzmann machines;**

## I. Introduction

Similarity search is a core problem in various areas such as machine learning, information retrieval, and data mining to name a few. For example, content-based image retrieval (CBIR) takes an image as a query and returns its nearest neighbors, computing similarity between visual descriptors (features) of the query and of images in database. A naive solution to nearest neighbor search is linear scan, but this approach is not scalable in practical applications. Approximate nearest neighbor search such as tree-based methods, is successful for low-dimensional data, but its performance is not satisfactory for high-dimensional data and does not guarantee faster search compared to linear scan [1]. Most of visual descriptors, such as SIFT [2], GIST [3], and HOG [4], constitute high-dimensional vectors, so tree-based space partition approach is not preferred in CBIR applications.

Hashing refers to methods for embedding high-dimensional data into a low-dimensional Hamming space such that similar objects are indexed by binary codes with small Hamming distances. Hashing can be categorized into data-independent and data-dependent methods. A notable data-independent method is locality sensitive hashing (LSH) [1], [5] where random projections followed by rounding are used to generate binary codes. The performance of LSH is not satisfactory when short binary codes are used [6]. Data-dependent hashing methods learn binary codes from a set of data to compactly index them. Learning hash functions can be done in unsupervised [7], supervised [8], or semi-supervised [9], [10] manner. Notable data-dependent hashing methods include: (1) spectral hashing [7] where a subset of eigenvectors of the Laplacian of the similarity graph is rounded to determine binary codes; (2) semantic hashing [8] where multi-layer networks are used to learn a non-linear mapping between input data and binary code bits.

Most of existing data-dependent hashing methods exploit only single representation of objects to learn hash functions, referred to as *single-view hashing*. In practice, however, images are often represented by different visual descriptors such as GIST [3], HOG [4], SIFT [2], and so on. These visual descriptors have their own characteristics and could be complementary to each other's strength and weakness. Thus, it is desirable to incorporate these heterogenous visual descriptors into learning hash functions, leading to *multi-view hashing*. Recently spectral hashing was extended to multi-view hashing where a linear sum of view-specific similarity matrices [11] was exploited. This is limited to a shallow model and considers linear hash functions followed by quantization to determine binary codes.

In this paper we present a deep network with a set of view-specific hidden nodes and a set of shared hidden nodes, that yields binary codes at its top layer nodes. There has been recent work on deep networks to handle multi-modal data [12] or designed for multi-view learning [13]. However, to our best knowledge, deep learning has not been exploited for multi-view hashing yet. This paper presents the first work on multi-view hashing using deep networks.

## II. Related Work

We briefly review spectral hashing [7] and its multi-view extension [11] Suppose that we are given a set of $N$ instances with $K$ different views, $\{\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(K)}\}_{i=1}^N$, where $\boldsymbol{x}_i^{(k)} \in \mathbb{R}^{D_k}$ correspond to visual descriptors. Then, the view-specific data matrix is defined as $\boldsymbol{X}^{(k)} = [\boldsymbol{x}_1^{(k)}, \ldots, \boldsymbol{x}_N^{(k)}] \in \mathbb{R}^{D_k \times N}$. We denote by $\boldsymbol{y}_i^{(k)} \in \{-1, +1\}^M$ a binary code of length $M$ associated with

$\boldsymbol{x}_i^{(k)}$. Then the binary code matrix is given by $\boldsymbol{Y}^{(k)} = [\boldsymbol{y}_i^{(k)}, \ldots, \boldsymbol{y}_N^{(k)}] \in \mathbb{R}^{M \times N}$. For single view hashing, the binary code matrix is represented by $\boldsymbol{Y}$ (without the superscript) and for multi-view hashing an integrated binary code matrix is denoted by $\boldsymbol{Y}^* = \{\boldsymbol{y}_i^*\}_{i=1}^N \in \mathbb{R}^{M \times N}$ which is expected to capture the average similarities between instances across views.

Spectral hashing [7] determines similarity-preserving compact binary codes by enforcing the average Hamming distance between similar neighbors to be minimized, and also requiring the codes of length $M$ to be uncorrelated and balanced. Thus, spectral hashing involves the following optimization:

$$\begin{aligned} \underset{\boldsymbol{Y}}{\arg\min} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} S_{i,j}\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2, \\ \text{subject to} \quad & \boldsymbol{Y} \in \{-1,+1\}^{M \times N}, \\ & \boldsymbol{Y}\boldsymbol{1}_N = \boldsymbol{0},\ \frac{1}{N}\boldsymbol{Y}\boldsymbol{Y}^\top = \boldsymbol{I}_M, \end{aligned} \tag{1}$$

where $S_{i,j}$ is the similarity between instances $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $\|\boldsymbol{y}_i\|_2$ is the Euclidean norm of binary code vector $\boldsymbol{y}_i$, $\boldsymbol{I}_M \in \mathbb{R}^{M \times M}$ denotes the identity matrix, and $\boldsymbol{1}_N \in \mathbb{R}^N$ is the vector of all ones. The problem is relaxed by discarding binary constraints $\boldsymbol{y}_i \in \{+1,-1\}^M$, so that rounding a subset of eigenvectors of the graph Laplacian of the similarity graph leads to binary codes for spectral hashing.

CHMIS-AW [11] takes a linear sum of view-specific similarities $S_{i,j}^* = \sum_k S_{i,j}$, which is plugged into the spectral hashing framework (1). For out-of-sample extension in CHMIS-AW, binary codes of unseen examples are determined by a convex combination of linear hash functions, i.e., $\sum_{k=1}^K \beta_k \boldsymbol{W}^{(k)\top}\boldsymbol{x}_i^{(k)}$ with $\sum_{k=1}^K \beta_k = 1$. Embedding matrices $\boldsymbol{W}^{(k)}$ and mixing coefficients $\beta_k$ are estimated by solving a regularized regression problem. $\boldsymbol{Y}^*$ and $\{\boldsymbol{W}^{(k)}, \beta_k\}_{k=1}^K$ are determined in an alternative fashion.

## III. Deep Multi-View Hashing

In this section we present the main contribution, *deep multi-view hashing* (DMVH), as shown in Fig. 1, which has a few unique features that highlight advantages: (1) Existing methods for multi-view hashing are limited to shallow models, while DMVH is expected to benefit from deep architecture; (2) Semantic hashing [8] builds a deep belief network for hashing, but it is limited to single view, while DMVH is able to manage multiple views; (3) DMVH consists of shared hidden nodes as well as view-specific hidden nodes to capture both shared and view-specific characteristics, while most of existing multi-view learning methods, including canonical correlation analysis [14] and dual wing harmonium (DWH) [15], assume that all views are completely correlated, which are captured by shared hidden nodes. Similar idea was used in shallow models [16], [17] and deep networks [13], but to our best knowledge it was not exploited for hashing yet.

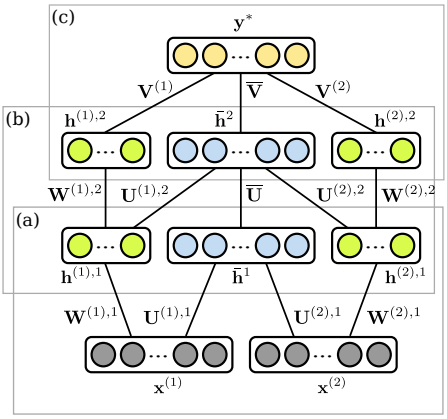



Figure 1. Graphical model for 4-layer DMVH.

### A. Model

The 4-layer DMVH model is shown in Fig. 1, where: (a) The bottom layer represents the visible nodes $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ with two different views, both of which are connected to the *shared hidden nodes* $\overline{\boldsymbol{h}}^1$ and each of which is connected corresponding *view-specific hidden nodes* $\boldsymbol{h}^{(1),1}$ and $\boldsymbol{h}^{(2),1}$ in the first hidden layer; (b) In the second hidden layer, the shared hidden nodes $\overline{\boldsymbol{h}}^2$ also further capture the common characteristics across hidden nodes in the first layer and the view-specific hidden nodes $\boldsymbol{h}^{(1),2}$ and $\boldsymbol{h}^{(2),2}$ are connected to corresponding hidden nodes in the first hidden layer; (c) The hidden nodes $\boldsymbol{y}^*$ in the top layer, which are connected to all the hidden nodes in the second hidden layer, represent the integrated binary code associated with $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$.

In order to handle both continuous and discrete variables, we choose the independent marginal distributions for visible nodes $\boldsymbol{x}^{(k)}$, view-specific hidden nodes $\boldsymbol{h}^{(k)}$, and shared hidden nodes $\overline{\boldsymbol{h}}$, from the exponential family, as in [18]:

$$\begin{aligned} p(\boldsymbol{x}^{(k)}) &= \prod_i \exp\Big\{\sum_{i,a} \xi_{i,a}^{(k)} f_{i,a}^{(k)}(x_i^{(k)}) - A_i^{(k)}(\{\xi_{i,a}^{(k)}\})\Big\}, \\ p(\boldsymbol{h}^{(k)}) &= \prod_j \exp\Big\{\sum_{j,b} \lambda_{j,b}^{(k)} g_{j,b}^{(k)}(h_j^{(k)}) - B_j^{(k)}(\{\lambda_{j,b}^{(k)}\})\Big\}, \\ p(\overline{\boldsymbol{h}}) &= \prod_j \exp\Big\{\sum_{j,b} \overline{\lambda}_{j,b}\overline{g}_{j,b}(\overline{h}_j) - \overline{B}_j(\{\overline{\lambda}_{j,b}\})\Big\}, \end{aligned}$$

where $\{\xi_{i,a}^{(k)}\}$, $\{\lambda_{j,b}^{(k)}\}$, $\{\overline{\lambda}_{j,b}\}$ are natural parameters, $\{f_{i,a}^{(k)}\}$, $\{g_{j,b}^{(k)}\}$, $\{\overline{g}_{j,b}\}$ are sufficient statistics, and $\{A_i^{(k)}\}$, $\{B_j^{(k)}\}$, $\{\overline{B}_j\}$ are log-partition functions. We illustrate random fields to describe each inter-layer model, denoted by (a), (b), (c) in Fig. 1, which are used to compute log-likelihoods.

*1) Model (a):* We assume Bernoulli distributions over hidden nodes:

$$p(\boldsymbol{h}^{(k),1}) = \prod_j \text{Bern}(h_j^{(k),1} \,|\, \sigma(\lambda_j^{(k),1})) = \prod_j \exp\left\{\lambda_j^{(k),1} h_j^{(k),1} - \log\left(1+\exp\left\{\lambda_j^{(k),1}\right\}\right)\right\},$$

$$p(\overline{\boldsymbol{h}}^1) = \prod_j \text{Bern}(\overline{h}_j^1 \,|\, \sigma(\overline{\lambda}_j^1)) = \prod_j \exp\left\{\overline{\lambda}_j^1 \overline{h}_j^1 - \log\left(1+\exp\left\{\overline{\lambda}_j^1\right\}\right)\right\},$$

where $\sigma(\cdot)$ is the logistic function and the natural parameters $\lambda_j^{(k),1} = \{\lambda_{j,b}^{(k),1}\}$ and $\overline{\lambda}_j^1 = \{\overline{\lambda}_{j,b}^1\}$ become log-odds. Sufficient statistics are given by $h_j^{(k),1} = \{g_{j,b}^{(k),1}(h_j^{(k),1})\}$ and $\overline{h}_j^1 = \{\overline{g}_{j,b}^1(\overline{h}_j^1)\}$. For visible nodes, we use Gaussian distributions with unit variance for real-valued descriptors and Bernoulli distributions for binary-valued descriptors:

$$p(\boldsymbol{x}^{(k)}) = \begin{cases} \prod_i \text{Bern}(x_i^{(k)} \,|\, \sigma(\xi_i^{(k)})), & \text{for binary,} \\ \prod_i \mathcal{N}(x_i^{(k)} \,|\, \xi_i^{(k)}, 1), & \text{for real-valued.} \end{cases}$$

Gaussian distribution $\mathcal{N}(x_i^{(k)} \,|\, \xi_i^{(k)}, 1)$ is specified by two pairs of natural parameters and sufficient statistics, and a log-partition function:

$$\{\xi_{i,a}^{(k)}\} = [\xi_i^{(k)}, -1/2]^\top, \quad \{f_{i,a}^{(k)}\} = [x_i^{(k)}, (x_i^{(k)})^2]^\top,$$
$$A_i^{(k)}(\{\xi_{i,a}^{(k)}\}) = -((\xi_i^{(k)})^2 + \log 2\pi)/2.$$

With these marginal distributions, we couple visible and hidden variables in the log-domain by introducing quadratic interaction terms, leading to the following random filed:

$$\begin{aligned} p(\{\boldsymbol{x}^{(k)}\}, \{\boldsymbol{h}^{(k),1}\}, \overline{\boldsymbol{h}}^1) \propto \exp\Big\{ & \sum_{k,i,a} \xi_{i,a}^{(k)} f_{i,a}^{(k)}(x_i^{(k)}) + \sum_j \overline{\lambda}_j^1 \overline{h}_j^1 + \sum_{k,j} \lambda_j^{(k),1} h_j^{(k),1} \\ & + \sum_{k,i,a,j} W_{i,a,j}^{(k),1} f_{i,a}^{(k)}(x_i^{(k)}) h_j^{(k),1} + \sum_{k,i,a,j} U_{i,a,j}^{(k),1} f_{i,a}^{(k)}(x_i^{(k)}) \overline{h}_j^1 \Big\}. \end{aligned}$$

Since the model is a bipartite graph, between-layer conditional distributions are represented as products of distributions of individual nodes, leading to the conditional distributions given by

$$\begin{aligned} p(x_i^{(k)} | \{\boldsymbol{h}^{(k),1}\}, \overline{\boldsymbol{h}}^1) &\propto \exp\Big\{\sum_a \widetilde{\xi}_{i,a}^{(k)} f_{i,a}^{(k)}(x_i) - A_i^{(k)}(\{\widetilde{\xi}_{i,a}^{(k)}\})\Big\}, \\ p(h_j^{(k),1} | \{\boldsymbol{x}^{(k)}\}, \overline{\boldsymbol{h}}^1) &\propto \exp\Big\{\widetilde{\lambda}_j^{(k),1} h_j^{(k),1} - B_j^{(k),1}(\{\widetilde{\lambda}_j^{(k),1}\})\Big\}, \\ p(\overline{h}_j^1 | \{\boldsymbol{x}^{(k)}\}, \{\boldsymbol{h}^{(k),1}\}) &\propto \exp\Big\{\widetilde{\overline{\lambda}_j^1} h_j^{(k),1} - \overline{B}_j^1(\{\widetilde{\overline{\lambda}_j^1}\})\Big\}. \end{aligned} \quad (2)$$

where $\widetilde{\xi}_{i,a}^{(k)} = \xi_{i,a}^{(k)} + \sum_j W_{i,a,j}^{(k),1} h_j^{(k)} + \sum_j U_{i,a,j}^{(k),1} \overline{h}_j^1$, $\widetilde{\lambda}_j^{(k),1} = \lambda_j^{(k),1} + \sum_{i,a} W_{i,a,j}^{(k),1} f_{i,a}^{(k)}(x_i^{(k)})$, and $\widetilde{\overline{\lambda}}_j^1 = \overline{\lambda}_j^1 + \sum_{i,a,k} U_{i,a,j}^{(k),1} f_{i,a}^{(k)}(x_i^{(k)})$ are shifted parameters.

*2) Model (b):* DMVH improves the limited representation power of 'Model (a)' by stacking an additional hidden layer on it. The second hidden layer is also composed of shared hidden nodes $\overline{\boldsymbol{h}}^2$ and view-specific hidden nodes $\boldsymbol{h}^{(k),2}$, which are connected to hidden nodes in the first hidden layer, in order to form higher-level representation (see (Fig. 1-(b)).

Assuming Bernoulli distributions for all visible and hidden nodes, the joint distribution is given by

$$\begin{aligned} p(\{\boldsymbol{h}^{(k),1}\}, \overline{\boldsymbol{h}}^1, \{\boldsymbol{h}^{(k),2}\}, \overline{\boldsymbol{h}}^2) \propto \exp\Big\{ & \sum_{k,i} \lambda_i^{(k),1} h_i^{(k),1} + \sum_i \overline{\lambda}_i^1 \overline{h}_i^1 + \sum_{k,j} \lambda_j^{(k),2} h_j^{(k),2} \\ & + \sum_j \overline{\lambda}_j^2 \overline{h}_j^2 + \sum_{k,i,j} W_{i,j}^{(k),2} h_i^{(k),1} h_j^{(k),2} + \sum_{k,i,j} U_{i,j}^{(k),2} h_i^{(k),1} \overline{h}_j^2 \\ & + \sum_{i,j} \overline{U}_{i,j}^2 \overline{h}_i^1 \overline{h}_j^2 \Big\}. \end{aligned}$$

We can repeatedly stack this structure in the middle of DMVH to build a desired number of layers. For example, stacking twice gives us 5-layer DMVH and omitting this structure results in 3-layer DMVH.

*3) Model (c):* The top two layers in DMVH combine high-level representations $\{\boldsymbol{h}^{(1),2}, \ldots, \boldsymbol{h}^{(K),2}, \overline{\boldsymbol{h}}^2\}$ into a set of integrated hash codes $\boldsymbol{y}^*$ (Fig. 1-(c)). Since hash codes are binary, the output node $y_j^*$ (which is also hidden) is assumed to follow Bernoulli distribution with mean $\sigma(\psi_j)$:

$$p(\boldsymbol{y}^*) = \prod_j \text{Bern}(y_j^* \,|\, \sigma(\psi_j)).$$

Then, the joint distribution over $(\{\boldsymbol{h}^{(k),2}\}, \overline{\boldsymbol{h}}^2, \boldsymbol{y}^*)$ is given by

$$\begin{aligned} p(\{\boldsymbol{h}^{(k),2}\}, \overline{\boldsymbol{h}}^2, \boldsymbol{y}^*) \propto \exp\Big\{ & \sum_{k,i} \lambda_i^{(k),2} h_i^{(k),2} + \sum_i \overline{\lambda}_i^2 \overline{h}_i^2 + \sum_j \psi_j y_j^* \\ & + \sum_{k,i,j} V_{i,j}^{(k)} h_i^{(k),2} y_j^* + \sum_{i,j} \overline{V}_{i,j} \overline{h}_i^2 y_j^* \Big\}. \end{aligned}$$

All these three models together constitute the 4-layer DMVH model. Our DMVH model is different from existing multi-view deep networks [12], [13] in many aspects. While existing models learn high-level features separately for each view and fuse them in the top layer, DMVH learns view-specific and shared representations across views in every layers to capture the partial correlations in multi-view data.

### *B. Training*

Training procedure of DMVH is similar to those of other deep networks. We first pre-train each layer in a greedy, layer-by-layer approach. We start by training parameters for the bottom two layers, then fix the learned parameters. Then we repeat the process for the higher two layers, until we reach the top layer. We train each two layers of DMVH by maximizing their expected log-likelihood $\mathcal{L}$ derived from the

joint distribution. For example, the expected log-likelihood of the bottom two layers of DMVH is as below:

$$\begin{aligned}\mathcal{L} &= \langle \log p(\{\boldsymbol{x}^{(k)}\})\rangle_+ \\ &= \left\langle \log \textstyle\sum_{\{\boldsymbol{h}^{(k),1}\},\overline{\boldsymbol{h}}^1} p(\{\boldsymbol{x}^{(k)}\}|\{\boldsymbol{h}^{(k),1}\},\overline{\boldsymbol{h}}^1)\right\rangle_+,\end{aligned}$$

where $\langle\cdot\rangle_+$ is expectation over data distribution. Derivation of likelihood of other layers is done in a similar way.

As exact calculation of the gradients of $\mathcal{L}$ requires summation or integration over the model distribution $p(\{\boldsymbol{x}^{(k)}\},\{\boldsymbol{h}^{(k),1}\},\overline{\boldsymbol{h}}^1)$, we need to do some approximation. Contrastive divergence [19] learning approximates model distribution by initializing visible nodes with data distribution, and then alternatively sampling from conditional distributions of visible and hidden nodes derived in (2). The gradient of $\mathcal{L}$ over the parameters of bottom two layers of DMVH is as follows:

$$\begin{aligned}\partial\mathcal{L}/\partial\boldsymbol{W}^{(k),1}_{i,a,j} &= \langle f^{(k)}_{i,a}B^{(k),1'}_j(\widetilde{\lambda}^{(k),1}_j)\rangle_+ - \langle f^{(k)}_{i,a}B^{(k),1'}_j(\widetilde{\lambda}^{(k),1}_j)\rangle_-,\\ \partial\mathcal{L}/\partial\boldsymbol{U}^{(k),1}_{i,a,j} &= \langle f^{(k)}_{i,a}\overline{B}^{1'}_j(\widetilde{\overline{\lambda}^1_j})\rangle_+ - \langle f^{(k)}_{i,a}\overline{B}^{1'}_j(\widetilde{\overline{\lambda}^1_j})\rangle_-,\\ \partial\mathcal{L}/\partial\xi^{(k)}_{i,a} &= \langle f^{(k)}_{i,a}\rangle_+ - \langle f^{(k)}_{i,a}\rangle_-,\\ \partial\mathcal{L}/\partial\lambda^{(k),1}_j &= \langle B^{(k),1'}_j(\widetilde{\lambda}^{(k),1}_j)\rangle_+ - \langle B^{(k),1'}_j(\widetilde{\lambda}^{(k),1}_j)\rangle_-,\\ \partial\mathcal{L}/\partial\overline{\lambda}^1_j &= \langle \overline{B}^{1'}_j(\widetilde{\overline{\lambda}^1_j})\rangle_+ - \langle \overline{B}^{1'}_j(\widetilde{\overline{\lambda}^1_j})\rangle_-,\end{aligned}$$

where $\langle\cdot\rangle_-$ is expectation over (approximated) model distribution, $f^{(k)}_{i,a}$ stands for $f^{(k)}_{i,a}(x^{(k)}_i)$, and $B^{(k),1'}_j = \partial B^{(k),1}_j/\partial\lambda^{(k),1}_j$ and $\overline{B}^{1'}_j = \partial\overline{B}^1_j/\partial\overline{\lambda}^1_j$ are derivatives of log-partition functions over parameters.

We also add additional penalty terms to the log-likelihood separate shared and view-specific features and enforce sparse hidden node activation. First, an orthogonalization term ensures that shared and view-specific hidden nodes do not contain similar features. We use a pairwise sum of cosine distances between columns of view-specific features $\boldsymbol{W}^{(k),1}$ and shared features $\boldsymbol{U}^{(k),1}$:

$$\mathcal{L}^{(k)}_{orth} = \sum_{i,j}\frac{\boldsymbol{w}^{(k),1\top}_i\boldsymbol{u}^{(k),1}_j}{\|\boldsymbol{w}^{(k),1}_i\|\|\boldsymbol{u}^{(k),1}_j\|}.$$

In addition, we enforce sparse activation of hidden nodes by setting their activation rate $\rho$ to a low value for numerical stability. For the output nodes, we set the activation rate to $0.5$ make sure each hash bit bisects the whole dataset:

$$\begin{aligned}\mathcal{L}^{(k)}_{sparsity} &= \textstyle\sum_j(\rho - \mathbb{E}[h^{(k),1}_j|\{\boldsymbol{x}^{(l)}\}])^2,\\ \overline{\mathcal{L}}_{sparsity} &= \textstyle\sum_j(\rho - \mathbb{E}[\overline{h}^1_j|\{\boldsymbol{x}^{(k)}\}])^2,\end{aligned}$$

By summing the expected likelihood, weight orthogonalization terms and sparsity penalty, we get the final objective function for pre-training.

$$\mathcal{L}_{DMVH} = \mathcal{L} - \sum_k(\alpha\mathcal{L}^{(k)}_{orth} + \beta\mathcal{L}^{(k)}_{sparsity}) - \beta\overline{\mathcal{L}}_{sparsity}.$$

Training other layers is done in a similar way. After pre-training each layers of DMVH, we fine-tune DMVH using objective function of nonlinear neighborhood component analysis (NCA) [20] with backpropagation algorithm:

$$\mathcal{L}_{NCA} = \sum_{i,j\in\mathcal{C}_i} p_{i,j} = \sum_{i,j\in\mathcal{C}_i}\frac{\exp(-d(\boldsymbol{y}^*_i,\boldsymbol{y}^*_j))}{\sum_{k\neq i}\exp(-d(\boldsymbol{y}^*_i,\boldsymbol{y}^*_k))}.$$

Nonlinear NCA maximizes the probability $p_{i,j}$ that hash codes $\boldsymbol{y}^*_i$ and $\boldsymbol{y}^*_j$ with the same class $\mathcal{C}_i$ are close to each other. This objective function is suitable for supervised hashing as it maps hash codes of instances with same labels to have small Hamming distances.

## IV. Experiments

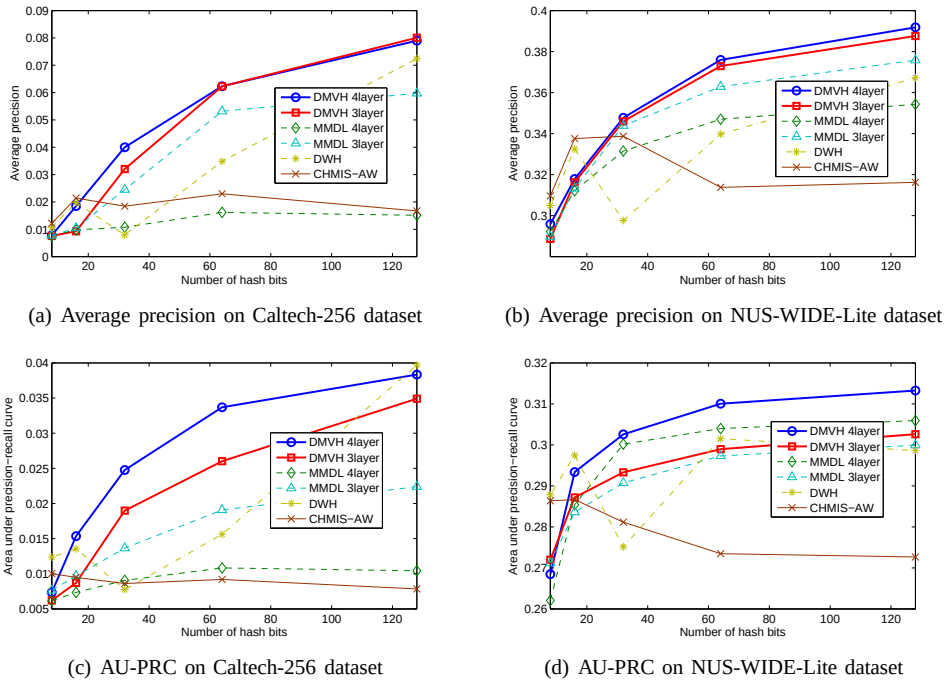
In this section, we performed image retrieval with hash codes learned by various hashing methods to investigate the usefulness of DMVH. Caltech-256 and NUS-WIDE-Lite dataset were used for our experiments. Caltech-256 [21] has 30,607 images with 256 classes. We extracted HoG [4] and GIST [3] descriptors from the dataset. 1000 samples were used as test querys, and the remaining were used for training. When building deep models for hashing this dataset, Gaussian distribution was assumed for both HOG and GIST descriptors. NUS-WIDE-Lite [22] contains 27,807+27,808 (training+test) images with tag annotations and 6 image descriptors. We used 5 of 6 image descriptors (excluding bag of visual words) and tag annotations. The dataset also consists of 81 different 'concepts', and these are used as labels for NCA fine-tuning. Gaussian distribution was assumed for 5 image descriptors, and Bernoulli distribution was chosen for tag annotations.

We compared existing deep networks including model for multi-modal deep learning (MMDL) and DWH with DMVH. We trained every model with layer-wise greedy pre-training followed by fine-tuning with NCA objective function (3). We pre-defined the number of nodes in DMVHs used in experiments. In 4-layer DMVHs, the second layer had 512+64 (shared+view-specific) hidden nodes. The third layer had 256+32 nodes. Then the top layer had node with the same number of hash code length. In 3-layer DMVHs, the middle layer had 512+64 nodes. For a fair comparison, we made sure that MMDL had similar number of parameters by assigning more hidden nodes for MMDL than DMVH. All deep models were trained with the same training parameters[1].

We also compared DMVH with CHMIS-AW, a shallow, multi-view hashing method based on spectral decomposition. We trained all hashing methods for hash code lengths $\{8,16,32,64,128\}$.

After training the models, we retrieved images from the training set whose hash code was within a hamming distance

[1]Models were trained for 50 epochs with batch size 100. Learning rate were set to 0.1, with momentum of 0.9. The parameters for orthogonalization penalty $\alpha$ and sparsity term $\beta$ were set to $\alpha = 0.0001$ and $\beta = 0.1$. For numerical stability, we set learning rate to 0.001 for pre-training bottom layers of deep networks.




(a) Average precision on Caltech-256 dataset

(b) Average precision on NUS-WIDE-Lite dataset

(c) AU-PRC on Caltech-256 dataset

(d) AU-PRC on NUS-WIDE-Lite dataset

Figure 2. Average precision (hamming radius=2) and area under precision-recall curve (AU-PRC) measured for hashing algorithms including DMVH, MMDL, DWH, and CHMIS-AW on Caltech-256 and NUS-WIDE-Lite datasets.

2 from the hash code of the query from test set. We increased the hamming distance boundary until we get more than 100 retrieved samples. After that, we calculated the precision by using labels as ground truth. The area under precision-recall curve was also computed for each hash code length. To increase the recall level, we increased hamming distance boundary used for retrieval.

On Caltech-256 dataset, DMVH models achieved the highest precision on every size of hash codes, followed by single-view deep network which uses more connection weights than it. DMVH scored the largest area under precision-recall curve also. DMVH also outperformed spectral hashing-based shallow multi-view hashing models, while still outperforming multimodal deep learning model. In both datasets, 4-layer DMVH gave better result than 3-layer model, empirically showing that the model is benefited from the advantage of using deep architecture for multi-view hashing. In contrast, adding more layer gave worse result on MMDL model (Fig. 2, 3).

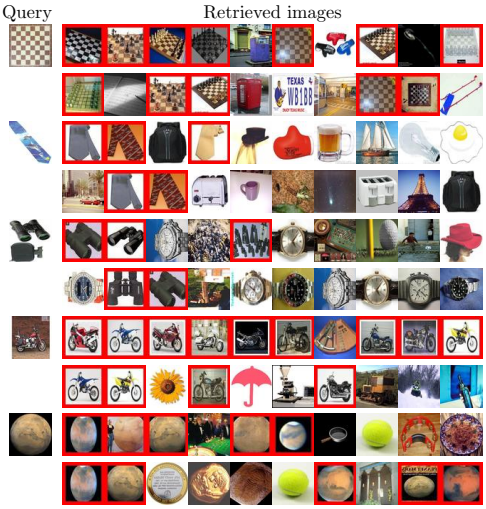



Figure 3. Qualitative comparison of 4-layer DMVH and 2-layer DWH on Caltech-256 dataset. For 5 query images (leftmost column), we retrieved 10 images using DMVH (upper rows) and DWH (lower rows). Retrieved images with the same label as the quary images were marked with red border.

## V. Conclusions

In this paper, we have proposed a deep network model for learning hash functions from partially correlated multi-view data. Each layer of the proposed model separates correlated and uncorrelated information, and propagates the higher-level shared representation generated by aggregating its inputs as well as the view-specific, uncorrelated information to upper layers. Then the model combines its outputs to generate a set of hash functions that effectively represents the partial correlation of multi-view inputs. By analyzing connection weights and hash codes learned from real-world image datasets, we have demonstrated that our model is capable of separating correlated and uncorrelated information on real-world datasets, and generates more semantically

meaningful hash codes than shallow ones. Moreover, our model has shown beneficial over the existing deep models and multi-view hashing algorithms based on spectral hashing on image retrieval experiments on image datasets with multiple descriptors.

**Acknowledgments:** This work was supported by NIPA-MSRA Creative IT/SW Research Project, NIPA ITRC Support Program (NIPA-2012-H0301-12-3002), the Converging Research Center Program funded by the Ministry of Education, Science, and Technology (2012K001343), MKE-NIPA "IT Consilience Creative Program" (C1515-1121-0003), and NRF World Class University Program (R31-10100).

## REFERENCES

[1] A. Gionis, P. Indyk, and R. Motawani, "Similarity search in high dimensions via hashing," in *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1999.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, 2005.

[5] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality sensitive hashing scheme based on $p$-stable distributions," in *Proceedings of the Annual ACM Symposium on Computational Geometry (SoCG)*, 2004.

[6] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, 2008.

[7] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 20. MIT Press, 2008.

[8] R. Salakhutdinov and G. Hinton, "Semantic hashing," in *Proceeding of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, 2007.

[9] J. Wang, S. Kumar, and S. F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, 2010.

[10] S. Kim and S. Choi, "Semi-supervised discriminant hashing," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, Vancouver, Canada, 2011.

[11] D. Zhang, F. Wang, and L. Si, "Composite hashing with multiple information sources," in *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Beijing, China, 2011.

[12] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, Bellevue, WA, 2011.

[13] Y. Kang and S. Choi, "Restricted deep belief networks for multi-view learning," in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, Athens, Greece, 2011.

[14] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with applications to learning methods," *Neural Computation*, vol. 16, pp. 2639–2664, 2004.

[15] E. P. Xing, R. Yan, and A. G. Hauptmann, "Mining associated text and images with dual-wing harmonium," in *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, Edinburgh, UK, 2005.

[16] H. Lee and S. Choi, "Group nonnegative matrix factorization for EEG classification," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Clearwater Beach, Florida, 2009.

[17] M. Salzmann, C. H. Ek, R. Urtasun, and T. Darrell, "Factorized orthogonal latent spaces," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy, 2010.

[18] M. Welling, M. Rosen-Zvi, and G. Hinton, "Exponential family harmoniums with an application to information retrieval," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 17. MIT Press, 2005.

[19] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[20] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, San Juan, Puerto Rico, 2007.

[21] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," Caltech, Tech. Rep., 2007.

[22] T. S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. 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 (CIVR)*, Santorini, Greece, 2009.