

Effective Multi-Modal Retrieval based on Stacked Auto-Encoders

Wei Wang[†], Beng Chin Ooi[†], Xiaoyan Yang[‡], Dongxiang Zhang[†], Yueting Zhuang[§]

[†]School of Computing, National University of Singapore, Singapore

[‡]Advanced Digital Sciences Center, Illinois at Singapore Pte, Singapore

[§]College of Computer Science, Zhejiang University, China

[†]{wangwei, ooibc, zhangdo}@comp.nus.edu.sg, [‡]xiaoyan.yang@adsc.com.sg, [§]yzhuang@zju.edu.cn

ABSTRACT

Multi-modal retrieval is emerging as a new search paradigm that enables seamless information retrieval from various types of media. For example, users can simply snap a movie poster to search relevant reviews and trailers. To solve the problem, a set of mapping functions are learned to project high-dimensional features extracted from data of different media types into a common low-dimensional space so that metric distance measures can be applied. In this paper, we propose an effective mapping mechanism based on deep learning (i.e., stacked auto-encoders) for multi-modal retrieval. Mapping functions are learned by optimizing a new objective function, which captures both intra-modal and inter-modal semantic relationships of data from heterogeneous sources effectively. Compared with previous works which require a substantial amount of prior knowledge such as similarity matrices of intra-modal data and ranking examples, our method requires little prior knowledge. Given a large training dataset, we split it into mini-batches and continually adjust the mapping functions for each batch of input. Hence, our method is memory efficient with respect to the data volume. Experiments on three real datasets illustrate that our proposed method achieves significant improvement in search accuracy over the state-of-the-art methods.

1. INTRODUCTION

The prevalence of social networking has significantly increased the volume and velocity of information shared on the Internet. A tremendous amount of data in various media types is being generated every day in the social networking systems, and images and video contribute the main bulk of the data. For instance, Twitter recently reported that over 340 million tweets were sent each day¹, while Facebook reported around 300 million photos were created each day². These data, together with other domain specific data, such as medical data, surveillance and sensory data, are

¹<https://blog.twitter.com/2012/twitter-turns-six>

²<http://techcrunch.com/2012/08/22/how-big-is-facebooks-data-2-5-billion-pieces-of-content-and-500-terabytes-ingested-every-day/>

Big Data that can be exploited for insights and contextual observations. However, effective retrieval of such huge amounts of media data from heterogeneous sources remains a big challenge.

In this paper, we study the problem of large-scale information retrieval from multiple modalities. Each modality represents one type of multimedia, such as text, image or video, and depending on the heterogeneity of data sources, we can have two following searches:

1. **Intra-modal search** has been extensively studied and widely used in commercial systems. Examples include web document retrieval via keyword queries and content-based image retrieval.
2. **Cross-modal search** enables users to explore more relevant resources from different modalities. For example, a user can use a tweet to retrieve relevant photos and videos from other heterogeneous data sources, or search relevant textual descriptions or videos by submitting an interesting image as a query.

There has been a long stream of research on multi-modal retrieval [27, 1, 15, 9, 25, 12, 26, 20]. These works share a similar query processing strategy which consists of two major steps. First, they learn a set of mapping functions to project the high-dimensional features from different modalities into a common low-dimensional latent space. Second, a multi-dimensional index for each modality in the metric space is built for efficient similarity retrieval. Since the second step is a classic kNN problem and has been extensively studied [6, 23], we shall focus on the optimization of the first step and propose a novel learning algorithm to find effective mapping functions.

We observe that most existing works, such as CVH [9], IMH [20], MLBE [25], CMSSH [1], and LSCMR [12], require a substantial amount of prior knowledge about the training data to learn effective mapping functions. Preparing prior knowledge in terms of large training dataset is labor-intensive, and due to manual intervention, the prepared knowledge may not be comprehensive in capturing the regularities (e.g., distribution or similarity relationships) of data. For example, MLBE and IMH require the similarity matrix of intra-modal data (one for each modality). It is usually collected by retrieving k (set by humans) nearest neighbors of every object in the original high-dimensional space and setting the corresponding entries in the similarity matrix as 1 (other entries as 0). Each training input of LSCMR is a ranking example, i.e., a list of objects ranked based on their relevance to the first one. Ranking examples are generated based on the labels of objects, which have to be created manually. CMSSH requires irrelevant inter-modal data pairs

that are also not available directly in most cases. In addition, many existing works (e.g., CVH, CMSSH, LCMH, IMH) have to load the whole training dataset into memory which becomes the bottleneck of training when the training dataset is too large to fit in memory.

To tackle the above issues, we propose a new mapping mechanism for multi-modal retrieval. Our mapping mechanism, called Multi-modal Stacked Auto-Encoders (MSAE), builds one Stacked auto-encoders (SAE) [22] for each modality and projects features from different media types into a common latent space. The stacked auto-encoders is a form of deep learning and has been successfully used in many unsupervised feature learning and classification tasks [16, 22, 4, 19]. The proposed MSAE has several advantages over existing approaches. First, the stacked structure of MSAE renders our mapping method (which is non-linear) more expressive than linear projections used in previous studies, such as CVH and LCMH. Second, compared with existing solutions, our method requires minimum amount of prior knowledge of the training data. Simple relevant pairs, e.g., images and their associated tags, are used as learning inputs. Third, the memory usage of our learning algorithm is independent of the training dataset size, and can be set to a constant.

For effective multi-modal retrieval, we need to learn a set of parameters in the MSAE such that the mapped latent features capture both inter-modal semantics and intra-modal semantics well. We design an objective function which takes the two requirements into consideration: the inter-modal semantics is preserved by minimizing the distance of semantic relevant inter-modal pairs in the latent space; the intra-modal semantics is captured by minimizing the reconstruction error (which will be discussed in detail in Section 2.2) of the SAE for each modality. By minimizing the reconstruction error, SAEs preserve the regularities (e.g., distribution) of the original features so that if the original features capture the semantics of data well, so do the latent features. When the original features of one modality are of low quality and cannot capture the intra-modal semantics, we decrease its weight of reconstruction error in the objective function. The intuition is that their intra-modal semantics can be preserved or even enhanced through inter-modal relationships with other modalities whose features are of high quality. Experiment results show that the latent features generated from our mapping mechanism are more effective than those generated by existing methods for multi-modal retrieval.

The main contributions of this paper are:

- We propose a novel mapping mechanism based on stacked auto-encoders to project high-dimensional feature vectors for data from different modalities into a common low-dimensional latent space, which enables effective and efficient multi-modal retrieval.
- We design a new learning objective function which takes both intra-modal and inter-modal semantics into consideration, and thus generates effective mapping functions.
- We conduct extensive experiments on three real datasets to evaluate our proposed mapping mechanism. Experimental results show that the performance of our method is superior to state-of-art methods.

The remaining of the paper is organized as follows. Problem statements and some backgrounds are provided in Section 2. After that, we describe the algorithm for training our mapping method in Section 3. Query processing is described in Section 4. Related works are discussed in Section 5. Finally we report our experiments in Section 6.

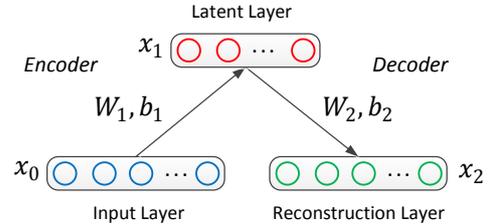


Figure 1: Auto-Encoder

2. PRELIMINARY

2.1 Problem Statements

In our data model, the database \mathbb{D} consists of objects from multiple modalities. For ease of presentation, we use images and text as two modalities to explain our idea, i.e., we assume that $\mathbb{D} = \mathbb{D}_I \cup \mathbb{D}_T$, even though our proposal is general enough to support any number of domains. Formally, an image is said to be relevant to a text document (e.g., a set of tags) if their semantic distance is small. Since there is no widely accepted measure to calculate the distance between an image and text, a common approach is to map images and text into the same latent space in which the two types of objects are comparable.

DEFINITION 1. Common Latent Space Mapping

Given an image feature vector $x \in \mathbb{D}_I$ and a text feature vector $y \in \mathbb{D}_T$, find two mapping functions $f_I : \mathbb{D}_I \rightarrow \mathbb{Z}$ and $f_T : \mathbb{D}_T \rightarrow \mathbb{Z}$ such that if x and y are semantically relevant, the distance between $f_I(x)$ and $f_T(y)$, denoted by $\text{dist}(f_I(x), f_T(y))$, is small in the common latent space \mathbb{Z} .

The common latent space mapping provides a unified approach to measuring distance of objects from different modalities. As long as all objects can be mapped into the same latent space, they become comparable. If the mapping functions f_I and f_T have been determined, the multi-modal search can then be transformed into the classic k NN problem, defined as following:

DEFINITION 2. Multi-Modal Search

Given a query object $Q \in \mathbb{D}_q$ ($q \in \{I, T\}$) and a target domain $\mathbb{D}_t \subset \mathbb{D}$ ($t \in \{I, T\}$), find a set $O \subset \mathbb{D}_t$ with k objects such that $\forall o \in O$ and $o' \in \mathbb{D}_t/O$, $\text{dist}(f_q(Q), f_t(o)) \geq \text{dist}(f_q(Q), f_t(o'))$.

Since both q and t have two choices, four types of queries can be derived, namely $Q_{q \rightarrow t}$ and $q, t \in \{I, T\}$. For instance, $Q_{I \rightarrow T}$ searches relevant text (e.g., tags) in \mathbb{D}_T given an image from \mathbb{D}_I . By mapping objects from high-dimensional feature space into low-dimensional latent space, queries can be efficiently processed using existing multi-dimensional indexes [6, 23]. Our goal is then to learn a set of effective mapping functions which preserve well both intra-modal semantics (i.e., semantic relationships within each modality) and inter-modal semantics (i.e., semantic relationships across modalities) in the latent space. The effectiveness is measured by the accuracy of multi-modal retrieval using latent features.

2.2 Auto-encoder

Auto-encoder and its variants have been widely used in unsupervised feature learning and classification tasks [16, 22, 4, 19]. It can be seen as a special neural network with three layers – the input layer, the latent layer, and the reconstruction layer (as shown in Figure 1). An auto-encoder contains two parts: (1) The encoder maps an input $x_0 \in \mathcal{R}^{d_0}$ to the latent representation (feature) $x_1 \in \mathcal{R}^{d_1}$ via a deterministic mapping f_e :

$$x_1 = f_e(x_0) = s_e(W_1^T x_0 + b_1) \quad (1)$$

where s_e is the activation function of the encoder, whose input is called the activation of the latent layer, and $\{W_1, b_1\}$ is the parameter set with a weight matrix $W_1 \in \mathbb{R}^{d_0 \times d_1}$ and a bias vector $b_1 \in \mathbb{R}^{d_1}$. (2) The decoder maps the latent representation x_1 back to a reconstruction $x_2 \in \mathcal{R}^{d_0}$ via another mapping function f_d :

$$x_2 = f_d(x_1) = s_d(W_2^T x_1 + b_2) \quad (2)$$

Similarly, s_d is the activation function of the decoder with parameters $\{W_2, b_2\}$, $W_2 \in \mathbb{R}^{d_1 \times d_0}$, $b_2 \in \mathbb{R}^{d_0}$. The input of s_d is called the activation of the reconstruction layer. Parameters are learned through back-propagation [17] by minimizing the loss function $\mathcal{L}(x_0, x_2)$ in Equation 3,

$$\mathcal{L}(x_0, x_2) = \mathcal{L}_r(x_0, x_2) + 0.5\xi(\|W_1\|_2^2 + \|W_2\|_2^2) \quad (3)$$

which consists of the reconstruction error $\mathcal{L}_r(x_0, x_2)$ and the L_2 regularization of W_1 and W_2 . By minimizing the reconstruction error, we require the latent features should be able to reconstruct the original input as much as possible. In this way, the latent features preserve regularities of the original data. The squared Euclidean distance is often used for $\mathcal{L}_r(x_0, x_2)$. Other loss functions such as negative log likelihood and cross-entropy are also used. The L_2 regularization term is a weight-decay which is normally added to the objective function to penalize large weights and reduce over-fitting [5]³. ξ is the weight decay cost, which is usually a small number (e.g., 10^{-4} in our experiments).

2.3 Stacked Auto-encoders (SAE)

The stacked auto-encoders (SAE) is a neural network with multiple layers of auto-encoders. It has been widely used as a deep learning method for dimensionality reduction [18] and feature learning [16, 22, 4, 19].

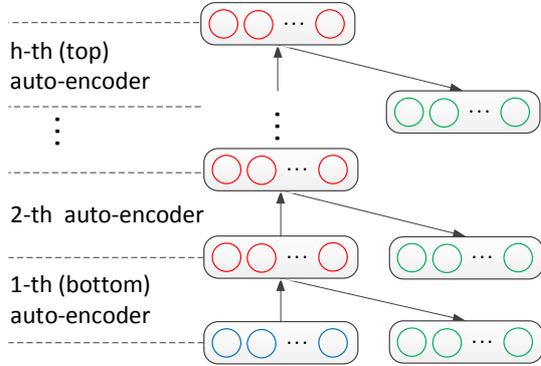


Figure 2: Stacked Auto-Encoders

As illustrated in Figure 2, there are h auto-encoders which are trained in a bottom-up and layer-wise manner. The input vectors (blue color in the figure) are fed to the bottom auto-encoder. After finishing training the bottom auto-encoder, the output latent representations are propagated to the higher layer. The sigmoid function or tanh function is typically used for the activation functions s_e and s_d . The same procedure is repeated until all the auto-encoders are trained. After such a pre-training stage, the whole neural network is

³For ease of presentation, we time all squared terms in this paper with a coefficient 0.5 to dismiss coefficients of the derivatives. E.g, the derivative of $\frac{\partial 0.5\|W_1\|_2^2}{\partial W_1} = W_1$

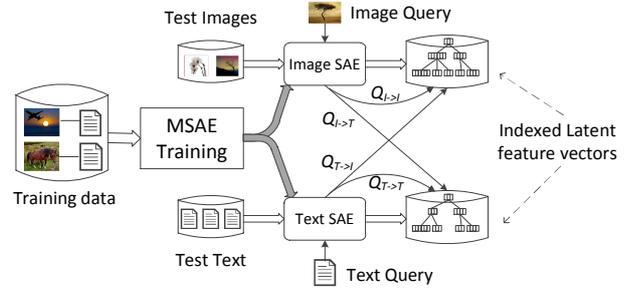


Figure 3: Process Flow of Learning and Query Processing.

fine-tuned based on a pre-defined objective. The latent layer of the top auto-encoder is the output of the stacked auto-encoders, which can be further fed into other applications, such as SVM for classification. The unsupervised pre-training can automatically exploit large amounts of unlabeled data to obtain a good weight initialization for the neural network than traditional random initialization.

2.4 Overview

Taking the image and text modalities as an example, we provide the process flow of the learning and query processing in Figure 3. Our learning algorithm MSAE is trained with simple image-text pairs. After training, we obtain one SAE for each modality. Test (database) images (resp. text), are mapped into the latent space through the image SAE (resp. text SAE). To support efficient query processing, we create an index for the latent feature vectors of each modality. All the steps are conducted offline. For real-time query processing, we map the query into the latent space through its modal-specific SAE, and return the kNNs using an appropriate index based on the query type. For example, the image index is used for queries against the image database, i.e., $Q_{I \rightarrow I}$ and $Q_{T \rightarrow I}$.

3. TRAINING

In this section, we design a two-stage training algorithm to learn the mapping functions of MSAE. A complete training process with image and text as two example modalities is illustrated in Figure 4.

In stage I, one SAE is trained for each modality. This training stage serves as the pre-training of stage II. As shown in step 1 and step 2 in the figure, we train each SAE independently with the objective to map similar features close to each other in the latent space. This pre-training stage provides a good initialization to the parameters in MSAE and improves the training performance. In stage II, we iterate over all SAEs, adjusting the parameters in one SAE at a time with the goal to capture both intra-modal semantics and inter-modal semantics. The learned MSAE projects semantically relevant objects close to each other in the latent space as shown by step 3 and 4 in Figure 4.

In the following, we first introduce the training algorithm of a single SAE, i.e., Stage I. Then we describe the training algorithm for MSAE, i.e., Stage II.

3.1 Single-Modal Training

Image Input We represent an image by a high dimensional real-valued vector. In the encoder, each input image is mapped to a latent vector using Sigmoid function as the activation function s_e (Equation 1). However, in the decoder, the Sigmoid activation function, whose range is $[0, 1]$, performs poorly on reconstruction because the input unit (referring to a dimension of the input) is not necessarily within $[0, 1]$. To solve the issue, we follow Hinton [5]

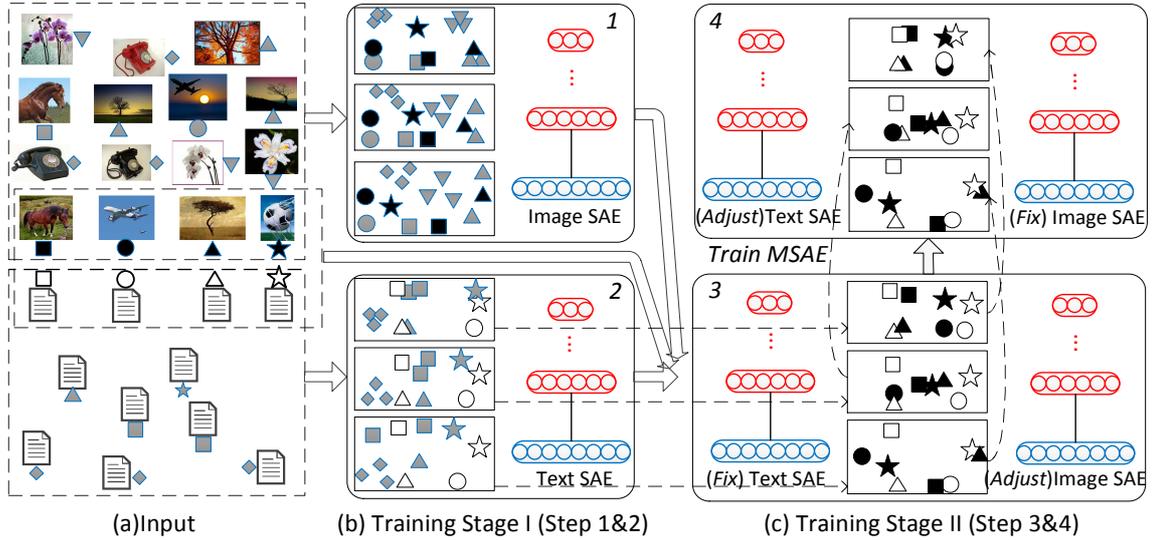


Figure 4: Flowchart of Training. Relevant images (or text) are associated with the same shape (e.g., ■).

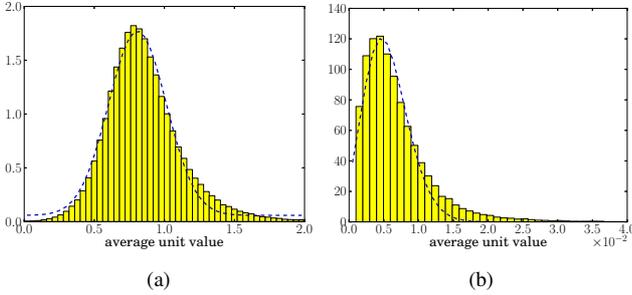


Figure 5: Distribution of image (5a) and text (5b) features extracted from NUS-WIDE training dataset (See Section 6). Each figure is generated by averaging the units for each feature vector, and then plot the histogram for all data.

and model the input unit as a linear unit with independent Gaussian noise. The reason is that the unit of image feature typically follows Gaussian distribution as shown in Figure 5a. Furthermore, the Gaussian noise term can be omitted if the input data is normalized with zero mean and unit variance. Consequently, we can use an identity function, denoted as s_d^I , for the activation function s_d in the decoder. We employ Euclidean distance to measure the reconstruction error for images, denoted as \mathcal{L}_r^I :

$$\mathcal{L}_r^I(x_0, x_2) = 0.5\|x_0 - x_2\|_2^2 \quad (4)$$

Text Input The text inputs are represented by word count vectors or tag occurrence vectors⁴. We adopt the Rate Adapting Poisson model [18] for the reconstruction because the histogram for text input unit generally follows Poisson distribution. Figure 5b shows the distribution of tags associated with the training images from NUS-WIDE training dataset. The activation function in the decoder is $x_2 = s_d^T(z_2) = l \times e^{z_2} / \sum_j e^{z_2j}$, where $l = \sum_j x_{0j}$ is the number of words in input text, x_1 is computed as in Equation 1 and $z_2 = W_2^T x_1 + b_2$. The probability of a reconstruction unit x_{2i}

⁴The value for each dimension indicates the corresponding tag appears or not.

being the same as the input unit x_{0i} is calculated as following:

$$p(x_{2i} = x_{0i}) = Pois(x_{0i}, x_{2i}) \quad (5)$$

where $Pois(n, \lambda) = e^{-\lambda} \lambda^n / n!$. Based on Equation 5, we define the reconstruction error as the negative log likelihood:

$$\mathcal{L}_r^T(x_0, x_2) = -\log \prod_i p(x_{2i} = x_{0i}) \quad (6)$$

Given a set of input feature vectors X^0 (each row is an image or text), the SAE consisting of h auto-encoders is trained by minimizing the following objective function:

$$\mathcal{L}(X^0) = \mathcal{L}_r(X^0, X^{2h}) + 0.5\xi \sum_{i=1}^h (\|W_1^i\|_2^2 + \|W_2^i\|_2^2) \quad (7)$$

where X^{2h} is the reconstruction of X^0 and $\mathcal{L}_r(X^0, X^{2h})$ is the average reconstruction error over examples in X^0 calculated using either Equation 4 or 6. The second term is the L_2 regularization of all parameter matrices. The objective function can be considered as an extension of Equation 3 to the stacked scenario. Generally speaking, if the reconstruction error is small, then the latent feature from the top auto-encoder would be able to reconstruct the original input well, and consequently, capture the regularities of the input data well. Therefore, if the feature vectors of two objects are similar in the original space, then they would be close in the latent space as shown by step 1 and step 2 in Figure 4.

The detailed procedure of training a single-modal SAE is shown in Algorithm 1. It consists of two major components: a layer-wise training stage (lines 1-3) which trains the auto-encoders from bottom to top with the objective function as Equation 3, and a fine-tuning stage (line 4) which adjusts all auto-encoders together to minimize Equation 7. The layer-wise training learns one single auto-encoder at a time, which can be seen as a special SAE with only one auto-encoder, i.e., $h = 1$. Its objective function is similar to that in the fine-tuning of SAE. Thus, they both call **trainNN** to perform the training task.

trainNN is an adaptation of the back-propagation algorithm [10] for training neural networks. By unfolding the SAE in to a neural network as shown in Figure 6, we can call **trainNN** to train it. As in [5], we divide the training dataset X^0 into mini-batches of the

Algorithm 1 $\text{trainSAE}(h, X^0, d)$

Input: h , height of SAE**Input:** X^0 , training data, one example per row**Input:** d , a sequence of dimensions for each layer**Output:** $\theta = \{\theta^i\}_{i=1}^h$, parameters of SAE

1. **for** $i = 1$ to h **do**
2. random init $\theta^i \leftarrow d_{i-1}, d_i$
3. $(\theta^i, X^i) = \text{trainNN}(1, X^{i-1}, \theta^i)$
4. $\theta \leftarrow \text{trainNN}(h, X^0, \theta)$

trainNN(h, X, θ)

1. **repeat**
 2. **for** batch B^0 in X **do**
 3. $Z, B = \text{fProp}(2h, B^0, \theta)$
 4. $\delta^{2h} = \frac{\partial \mathcal{L}(B^0)}{\partial Z^{2h}}$
 5. **bProp**($2h, \delta^{2h}, B, Z, \theta$) //(see Appendix)
 6. **until** converge
 7. **return** $\text{fProp}(h, X, \theta)$
-

same size (line 2). For example, given a dataset with 60,000 images, we can divide them into 600 batches, each with 100 images. For each mini-batch, we forward propagate the input to compute the value of each layer (**fProp** in line 3). Specifically, the latent layers (in Figure 6) are calculated according to Equation 1. The reconstruction layers are calculated according to Equation 2 except that of the bottom auto-encoder (i.e., the right most layer in Figure 6), which is modal dependent. For image (resp. text) SAE the s_d of the bottom auto-encoder (i.e., the right most s_d in Figure 6) is s_d^I (resp. s_d^T). In line 4, we calculate the partial derivative of the objective function \mathcal{L} w.r.t. the activation of the last layer. After that, we back-propagate the derivative to get gradients of parameters in each layer (**bProp** in line 5) and update them according to Stochastic Gradient Descent. The details of parameter updating are described in the Appendix section. In line 7, the latent features from the top auto-encoder are returned as the input for training upper auto-encoders.

3.2 Multi-Modal Training

Single modal training initializes one SAE for each modality with the objective to minimize the reconstruction error. The generated latent features thus preserve the regularities of the original features well. But it does not necessarily capture the semantics of the intra-modal data. If the original features do not capture the intra-modal semantics well, the latent features would fail to capture the intra-modal semantics. Moreover, inter-modal semantics is not involved in the single modal training, thus cannot be captured by the latent features. To preserve both intra-modal semantics and inter-modal semantics in the latent features, we combine all modalities together to learn an effective mapping mechanism.

The intuition of multi-modal training is as follows. On the one hand, in the learning objective function, we minimize the distance of latent features of semantic relevant inter-modal pairs. The learned mapping functions would then try to map semantic relevant inter-modal pairs into similar latent features. On the other hand, we minimize the reconstruction error for the modalities whose original features are of high quality in capturing intra-modal semantics. In this way, the latent features preserve the regularities of the original features well and thus captures semantics well. For modalities with low quality features, we assign small weights for their reconstruction error in the objective function. In this manner, the restriction of minimizing the reconstruction error is relaxed, while the restric-

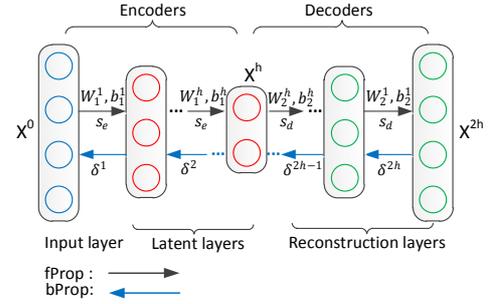


Figure 6: Unfolded Stacked Auto-Encoders

tion on minimizing the inter-modal distance is enhanced relatively. Consequently, the intra-modal semantics of low quality modalities can be preserved or even enhanced through their inter-modal relationships with the modalities of high quality features. For example, let x_0, y_0 be two semantic relevant objects from two different modalities, namely, x and y , where x 's feature is of high quality in capturing semantics while y 's feature is of low quality. If x_1 and y_1 are their latent features generated by minimizing the reconstruction error, then y_1 can preserve the semantics well, but x_1 is not as meaningful due to the low quality of x_0 . To solve this problem, we combine the inter-modal distance between x_1 and y_1 in the learning objective function and assign smaller weight for the reconstruction error of x_1 . The effect is the same as increasing the weight for minimizing the inter-modal distance with y_1 . As a result, the objective function will adjust the parameters to make the distance between x_1 and y_1 become smaller. In this way, the semantics of low quality x_1 is indirectly enhanced by the high quality feature y_1 .

With the above learning intuition, we define our objective function for multi-modal training as follows,

$$\mathcal{L}(X^0, Y^0) = \alpha \mathcal{L}_r^I(X^0, X^{2h}) + \beta \mathcal{L}_r^T(Y^0, Y^{2h}) + \mathcal{L}_d(X^h, Y^h) + \xi(\theta) \quad (8)$$

where X^0 (resp. Y^0) is a matrix for the image (resp. text) training feature vectors; each row of X^0 and Y^0 makes a semantic relevant inter-modal pair, e.g., an image and its associated tags; X^{2h} (resp. Y^{2h}) is the corresponding reconstruction matrix, which is calculated by forwarding the input through the unfolded image (resp. text) SAE as shown in Figure 7. X^h and Y^h are the latent feature matrices, which are the outputs of MSAE (see Equation 7). \mathcal{L}_r^I (see Equation 4) is the reconstruction error from image SAE, and \mathcal{L}_r^T (see Equation 6) is the reconstruction error from text SAE. \mathcal{L}_d is the distance function for latent features, which is the Euclidean distance (see Equation 4) in our implementation; similar to Equation 7, the last term $\xi(\theta)$ is the L_2 regularization of the parameter matrices involved in all SAEs.

α and β are the weights for the reconstruction error of image and text SAEs respectively, which are set according to the quality of the original features in capturing intra-modal semantics. After single modal training, we can test the quality of the original features for each modality on a validation dataset by performing intra-modal search against the latent features. The weight, e.g., α , is assigned with smaller value if the performance is worse, i.e., the original features are not good at capturing semantics of the data. In this manner, the weight of \mathcal{L}_d would increase relatively in terms of the image modality, which moves images close to their semantic relevant text in the latent space. Consequently, the latent features of images would gain more semantics from the text latent features. Since the dimensions of the latent space and the original space are

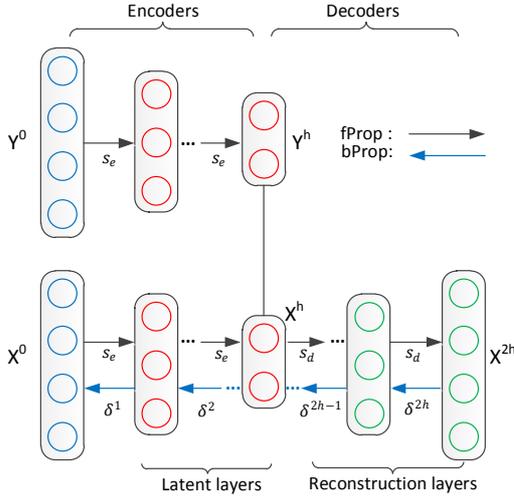


Figure 7: Unfolded Multi-modal Stacked Auto-Encoders

Algorithm 2 $\text{trainMSAE}(h, X^0, Y^0, \theta)$

Input: h , height of MSAE

Input: X^0, Y^0 , image and text input data

Input: $\theta=(\theta_X, \theta_Y)$, parameters of MSAE, initialized by **trainSAE**

Output: θ , updated parameters

1. **repeat**
2. **trainMNN**($h, X^0, Y^0, \theta_X, \theta_Y$)/train image SAE
3. **trainMNN**($h, Y^0, X^0, \theta_Y, \theta_X$)/train text SAE
4. **until** converge

trainMNN($h, X, Y, \theta_X, \theta_Y$)

Input: X , input data for the modality whose SAE is to be updated

Input: Y , input data for the modality whose SAE is fixed

Input: θ_X, θ_Y , parameters for the two SAEs.

1. **repeat**
 2. **for** batch (B_X^0, B_Y^0) in (X, Y) **do**
 3. $B_X, Z_X = \mathbf{fProp}(2h, B_X^0, \theta_X)$
 4. $B_Y, Z_Y = \mathbf{fProp}(h, B_Y^0, \theta_Y)$
 5. $\delta^{2h} = \frac{\partial \mathcal{L}(B_X^0, B_Y^0)}{\partial Z_X^{2h}}$
 6. $\delta^h = \mathbf{bProp}(h, \delta^{2h}, \{B_X^i\}_{i=h}^{2h}, \{Z_X^i\}_{i=h}^{2h}, \{\theta_X^i\}_{i=h}^{2h})$
 7. $\delta^h_+ = \frac{\partial \mathcal{L}_d(B_X^h, B_Y^h)}{\partial Z_X^h}$
 8. $\mathbf{bProp}(h, \delta^h, \{B_X^i\}_{i=0}^h, \{Z_X^i\}_{i=1}^h, \{\theta_X^i\}_{i=1}^h)$
 9. **until** converge
-

usually of different orders of magnitude, the scale of \mathcal{L}_r^I , \mathcal{L}_r^T and \mathcal{L}_d are different. α and β need to be scaled to make them comparable, i.e., within an order of magnitude.

Algorithm 2 shows the procedure of training multi-modal SAE (MSAE). Instead of adjusting both SAEs simultaneously, we iterate over SAEs, adjusting one SAE at a time with the other one fixed, as shown in lines 2-3. This is because the training of the neural networks is prone to local optimum. Tuning multiple SAEs simultaneously makes the training algorithm more difficult to converge. By adjusting only one SAE with the other one fixed, the training of MSAE turns to be much easier. Experiments show that one to two iterations is enough for converging. The convergence is monitored by performing multi-modal retrieval on a validation dataset.

trainMNN in Algorithm 2 adjusts the SAE for modality X with the SAE of modality Y fixed. It is an extension of **tranNN**, and

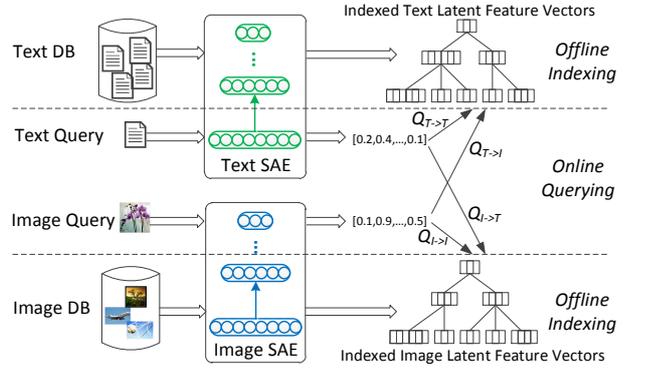


Figure 8: Illustration of Query Processing

thus adjusts parameters based on Stochastic Gradient Descent as well. We illustrate the training algorithm taking X being the image modality as an example. The unfolded image SAE is shown in Figure 7, which is exactly the same as that in Figure 6. Since the text SAE is fixed, $\mathcal{L}_r^T(Y^0, Y^{2h})$ is a constant in Equation 8. The reconstruction layers of the text SAE are then not involved in learning parameters of the image SAE. Hence, we do not show them in Figure 7. In line 3, we forward propagate the input through all layers of the image SAE including latent layers and reconstruction layers. But we do not calculate the reconstruction layers for text SAE in line 4 because the reconstructions are constant and have no contribution to updating the image SAE. To update the parameters of the decoders, we firstly calculate the partial derivative of the objective function, i.e., Equation 8, w.r.t. the activation of the last reconstruction layer. Next, we back-propagate the partial derivative to the top latent layer in line 6 calculating the gradients of parameters in each decoder and updating them. In line 7, the partial derivative of \mathcal{L}_d is combined. In line 8, **bProp** updates the parameters of each encoder according to their gradients.

4. QUERY PROCESSING

After training MSAE, each modality is associated with its own SAE whose parameters are already well learned. Given a set of heterogeneous data sources, high-dimensional features are extracted from each source and mapped into a low-dimensional latent space using the trained SAE. For example, if we have one million images, we first convert them into bag-of-words representations. Specifically, SIFT features are extracted from images and clustered into N bags. Each bag is considered a visual word and each image is represented by an N -dimensional vector. Our goal is to map the N -dimensional feature into a common latent space with m dimensions (m is normally set small, e.g., 16, 24 or 32). The mapping procedure is illustrated in Algorithm 3. Image input (resp. text input) is forwarded through encoders of the image SAE (resp. text SAE) to the top latent layer. Line 1 extracts parameters of the modal-specific encoders (see Figure 6). The actual mapping is conducted by **fProp** (see Algorithm 1) at line 2.

After the mapping, we create VA-File [23] to index the latent features (one index per modality). Given a query input, we check its media type and map it into the low-dimensional space through its modal-specific SAE as shown by Algorithm 3. Next, intra-modal and inter-modal searches are conducted against the corresponding index shown in Figure 8. For example, the task of searching relevant tags of one image, i.e., $Q_{I \rightarrow T}$, is processed by the index of the text latent vectors.

Algorithm 3 Inference(D)

Input: D , high-dimensional (image/text) query feature vectors
1. $\theta \rightarrow \{(W_1^1, b_1^1), (W_1^2, b_1^2), \dots, (W_1^h, b_1^h)\}$ //parameters of encoders
2. return $\mathbf{fProp}(h, D, \theta)$

To further improve the search efficiency, we convert the real-valued latent features into binary features, and search based on Hamming distance. The conversion is conducted using existing hash methods that preserve the neighborhood relationship based on Euclidean distance. For example, in our experiment, we choose Spectral Hashing [24], which converts real-valued vectors (data points) into binary codes with the objective to minimize the Hamming distance of data points who are close in the original Euclidean space.

However, the conversion from real-valued features to binary features trades off effectiveness for efficiency. Since there is information loss when real-valued data is converted to binaries, it affects the retrieval performance. We study the trade-off between efficiency and effectiveness on binary features and real-valued features in the experiment section.

5. RELATED WORK

The key problem of multi-modal retrieval is to find an effective mapping mechanism, which maps data from different modalities into a common latent space. An effective mapping mechanism would preserve both intra-modal semantics and inter-modal semantics well in the latent space, and thus generates good retrieval performance.

Linear projection has been studied to solve this problem [9, 20, 26]. Generally they try to find a linear projection matrix for each modality which maps semantic relevant data into similar latent vectors. However, if the distribution of the original data is non-linear, it would be hard to find a set of good projection matrices to make the latent vectors of relevant data close. CVH [9] extends the Spectral Hashing [24] to multi-modal data by finding a linear projection for each modality that minimizes the Euclidean distance of relevant data in the latent space. Similarity matrices for both inter-modal data and intra-modal data are required to learn a set of good mapping functions. IMH [20] learns the latent features of all training data firstly, which costs expensively. LCMH [26] exploits the intra-modal correlations by representing data from each modality using its distance to cluster centroids of the training data. Projection matrices are then learned to minimize the distance of relevant data (e.g., image and tags) from different modalities.

Other recent works include CMSSH [1], MLBE [25] and LSCMR [12]. CMSSH uses a boosting method to learn the projection function for each dimension of the latent space. However, it requires prior knowledge such as semantic relevant and irrelevant pairs. MLBE learns the latent features of all training data using a probabilistic graphic model firstly. Then it learns the latent features of queries based on their correlation with the training data. It does not consider the original features (e.g., image visual feature or text feature). Instead, only the correlation of data (both inter-similarity and intra-similarity matrices) are involved in the probabilistic model. However, the label of a query is usually not available in practice, which makes it impossible to obtain its correlation with the training data. LSCMR [12] learns the mapping functions with the objective to optimize the ranking criteria (e.g., MAP) directly. Ranking examples (a ranking example is a query and its ranking list) are needed for training. In our algorithm, we use simple relevant pairs

Table 1: The Statistics of Datasets

Dataset	NUS-WIDE	Wiki	Flickr1M
Total size	190,421	2,866	1,000,000
Training set	60,000	2,000	975,000
Validation set	10,000	366	6,000
Test set	120,421	500	6,000
Average Text Length	6	131	5

(e.g., image and its tags) as training input, thus no prior knowledge such as irrelevant pairs, similarity matrix, ranking examples and labels of queries, is needed.

Multi-modal deep learning [14, 21] extends Deep Learning to multi-modal scenario. [21] combines two Deep Boltzman Machines (DBM) (one for image, one for text) with a common latent layer to construct a Multi-modal DBM. [14] constructs a Bimodal deep auto-encoder with two deep auto-encoders (one for audio, one for video). Both two models aim to improve the classification accuracy of objects with features from multiple modalities. Thus they combine different features to learn a good (high dimensional) latent feature. In this paper, we aim to represent data with low-dimensional latent features to enable effective and efficient multi-modal retrieval, where both the query and database objects may have features from only one modality.

6. EXPERIMENTAL STUDY

This section provides an extensive performance study of our solution in comparison with the state-of-the-art methods: CVH [9], CMSSH [1] and LCMH [26]⁵. We examine both efficiency and effectiveness of our method including training overhead, query processing time and accuracy. All the experiments are conducted on CentOS 6.4 using CUDA 5.0 with NVIDIA GPU (GeForce GTX TITAN). The size of main memory is 64GB and GPU memory is 4GB. The code and hyper-parameter setting is available online⁶.

6.1 Datasets

We evaluate the performance on three benchmark datasets—NUS-WIDE [2], Wiki [15] and Flickr1M [7].

NUS-WIDE The dataset contains 269,648 images from Flickr, each associated with 6 tags in average. We refer to the image and its tags as an image-text pair. There are 81 ground truth concepts manually annotated for evaluation. Following previous works [11, 26], we extract 190,421 image-text pairs annotated with the most frequent 21 concepts and split them into three subsets for training, validation and test respectively. The size of each subset is shown in Table 1. For validation (resp. test), 100 (resp. 1000) queries are randomly selected from the validation (resp. test) dataset. Image and text features have been provided in the dataset [2]. For images, SIFT features are extracted and clustered into 500 visual words. Hence, an image is represented by a 500 dimensional bag-of-visual-words vector. Its associated tags are represented by a 1,000 dimensional tag occurrence vector.

Wiki This dataset contains 2,866 image-text pairs from Wikipedia’s featured articles. An article in Wikipedia contains multiple sections. The text information and its associated image in one section is considered as an image-text pair. Every image-text pair has

⁵The code and parameter configurations for CVH and CMSSH are available online at <http://www.cse.ust.hk/~dyyeung/code/mlbe.zip>; The code for LCMH is provided by the authors. Parameters are set according to the suggestions provided in the paper.

⁶<http://www.comp.nus.edu.sg/~wangwei/code>

a concept inherited from the article’s category (there are 10 categories in total). We randomly split the dataset into three subsets as shown in Table 1. For validation (resp. test), we randomly select 50 (resp. 100) pairs from the validation (resp. test) set as the query set. Images are represented by 128 dimensional bag-of-visual-words vectors based on SIFT feature. For text, we construct a vocabulary with the most frequent 1,000 words excluding stop words, and then represent one text section by 1,000 dimensional word count vector like [12]. The average number of words in one section is 131 which is much higher than that in NUS-WIDE. To avoid overflow in Equation 5 and smooth the text input, we normalize each unit x as $\log(x + 1)$ [18].

Flickr1M This dataset contains 1 million images associated with tags from Flickr. 25,000 of them are annotated with concepts (there are 38 concepts in total). The image feature is a 3,857 dimensional vector concatenated by SIFT feature, color histogram, and etc [21]. Like NUS-WIDE, the text feature is represented by a tag occurrence vector with 2,000 dimensions. All the image-text pairs without annotations are used for training. For validation and test, we randomly select 6,000 pairs with annotations respectively, among which 1,000 pairs are used as queries.

Before training, we use ZCA whitening [8] to normalize each dimension of image feature to have zero mean and unit variance.

Given a query, the *ground truth* is defined as: *if a result shares at least one common concept with the query, it is considered as a relevant result; otherwise it is irrelevant.*

6.2 Evaluation Metrics

Firstly, we study the effectiveness of the mapping mechanism. It is reflected by the effectiveness of the multi-modal search, i.e., $\mathbb{Q}_{q \rightarrow t}(q, t \in \{T, I\})$, using the mapped latent features⁷. Hence, we use the Mean Average Precision (MAP) [13], one of the standard information retrieval metrics, as the major effectiveness evaluation metric. Given a set of queries, we first calculate the Average Precision (AP) for each query,

$$AP(q) = \frac{\sum_{i=1}^R P(i)\delta(i)}{\sum_{j=1}^R \delta(j)}$$

where R is the size of the test dataset; $\delta(i) = 1$ if the i -th result is relevant, otherwise $\delta(i) = 0$; $P(k)$ is the precision of the result ranked at position k , which is the fraction of true relevant documents in the top k results. By averaging AP for all queries, we get the MAP score. The larger the MAP score, the better the search performance. In addition to MAP, we also measure the precision and recall of search tasks.

Besides effectiveness, we also evaluate the training overhead in terms of time cost and memory consumption. In addition, we report the evaluation on query processing time at last.

6.3 Visualization of Training Process

In this section we visualize the training process of MSAE using the NUS-WIDE dataset as an example to help understand the intuition of the training algorithm and the setting of the weight parameters, i.e., α and β . Our goal is to learn a set of mapping functions such that the mapped latent features capture both intra-modal semantics and inter-modal semantics well. Generally, the inter-modal semantics is preserved by minimizing the distance of the latent features of relevant inter-modal pairs. The intra-modal semantics is preserved by minimizing the reconstruction error of each SAE and through inter-modal semantics (see Section 3 for details).

⁷Without specifications, searches are conducted against real-valued latent features using Euclidean distance.

Firstly, following the training procedure in Section 3, we train a 4-layer image SAE with the dimension of each layer as $500 \rightarrow 128 \rightarrow 16 \rightarrow 2$ using Algorithm 1. Similarly, a 4-layer text SAE (the structure is $1000 \rightarrow 128 \rightarrow 16 \rightarrow 2$) is trained. Latent features of sampled images from the validation set are plotted in blue circles as shown in Figure 9a. Similarly, latent features of sampled text are plotted in white circles. This pre-training stage initializes SAEs to capture regularities of the original features of each modality in the latent features. On the one hand, the original features may be of low quality to capture intra-modal semantics. In such a case, the latent features would also fail to capture the intra-modal semantics. In fact, we evaluate the quality of the mapped latent features from each SAE by intra-modal search on the validation dataset. The MAP of the image intra-modal search is about 0.37, while that of the text intra-modal search is around 0.51. On the other hand, the SAEs are trained separately. Therefore, inter-modal semantics are not considered. We randomly pick 25 relevant image-text pairs and connect them with red lines in Figure 9b. We can see the latent features of most pairs are far away from each other, which indicates that the inter-modal semantics are not captured by these latent features. To solve the above problems, we resort to the multi-modal training by Algorithm 2. In the following figures, we only plot the distribution of these 25 pairs for easy of illustration.

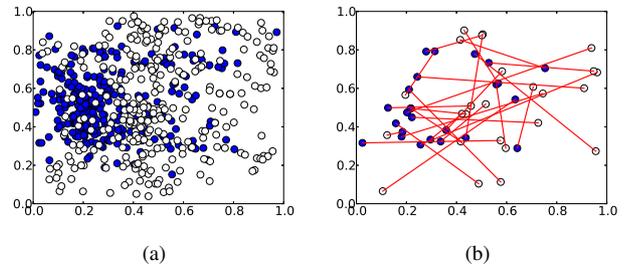
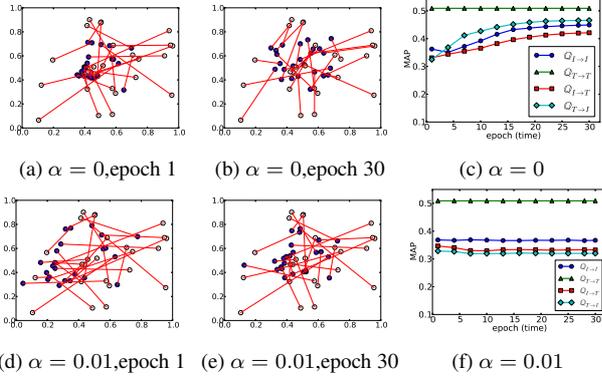
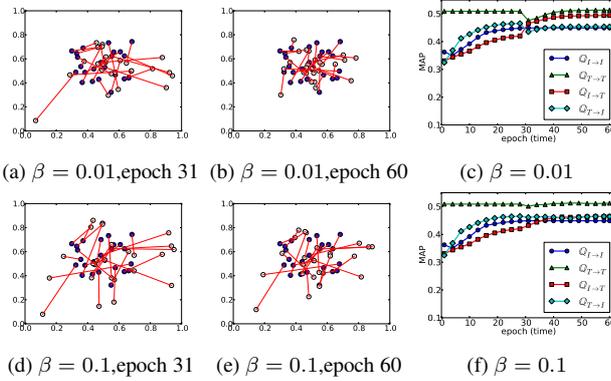


Figure 9: Latent Features (Blue circles are image latent features; White circles are text latent features)

Secondly, we adjust the image SAE with the text SAE fixed as line 2 of Algorithm 2 from epoch 1 to epoch 30. One epoch means one pass of the whole training dataset. Since the MAP of the image intra-modal search is worse than that of the text intra-modal search, according to the analysis in Section 3.2, we should use small α to decrease the weight of image reconstruction error \mathcal{L}_r^I in the objective function, i.e., Equation 8. To verify the correctness of the analysis, we compare the performance of two choices of α , namely $\alpha = 0$ and $\alpha = 0.01$. The first two columns of Figure 10 show the latent features generated by the image SAE after epoch 1 and epoch 30. Comparing Figure 10b and 10e, we can see that with smaller α , the image latent features are moved closer to their relevant text latent features. This is in accordance with Equation 8, where smaller α relaxes the restriction on the image reconstruction error, and in turn increases the weight for \mathcal{L}_d . By moving close to relevant text latent features, the image latent features gain more semantics. As shown in Figure 10c, the MAP curves keep increasing with the training going on and converge when inter-modal pairs are close enough. $\mathbb{Q}_{T \rightarrow T}$ does not change because the text SAE is fixed. Because image latent features are hardly moved close to the relevant text latent features when $\alpha = 0.01$ as shown in Figure 10d and 10e, the MAP curves do not increase in Figure 10f. We use the results with $\alpha = 0$ to continue the training procedure in the following section.

Table 2: Mean Average Precision on NUS-WIDE dataset

Task	$\mathbb{Q}_{I \rightarrow I}$				$\mathbb{Q}_{T \rightarrow T}$				$\mathbb{Q}_{I \rightarrow T}$				$\mathbb{Q}_{T \rightarrow I}$				
	Algorithm	LCMH	CMSSH	CVH	MSAE	LCMH	CMSSH	CVH	MSAE	LCMH	CMSSH	CVH	MSAE	LCMH	CMSSH	CVH	MSAE
Dimension of Latent Space L	16	0.353	0.355	0.365	0.417	0.373	0.400	0.374	0.498	0.328	0.391	0.359	0.447	0.331	0.337	0.368	0.432
	24	0.343	0.356	0.358	0.412	0.373	0.402	0.364	0.480	0.333	0.388	0.351	0.444	0.323	0.336	0.360	0.427
	32	0.343	0.357	0.354	0.413	0.374	0.403	0.357	0.470	0.333	0.382	0.345	0.402	0.324	0.335	0.355	0.435

Figure 10: Adjusting Image SAE with Different α Figure 11: Adjusting Text SAE with Different β

Thirdly, according to line 3 of Algorithm 2, we adjust the text SAE with the image SAE fixed from epoch 31 to epoch 60. We also compare two choices of β , namely 0.01 and 0.1. Figure 11 shows the snapshots of latent features and the MAP curves of each setting. From Figure 10b to 11a, which are two consecutive snapshots taken from epoch 30 and 31 respectively, we can see that the text latent features are moved much close to the relevant image latent features. It leads to the big changes at epoch 31 in Figure 11c. For example, $\mathbb{Q}_{T \rightarrow T}$ drops a lot. This is because the sudden move changes the intra-modal relationships of text latent features. Another big change happens on $\mathbb{Q}_{I \rightarrow T}$, which increases dramatically. The reason is that when we fix the text features from epoch 1 to 30, an image feature I is pulled to be close to (or nearest neighbor of) its relevant text feature T . However, T may not be the reverse nearest neighbor of I . In epoch 31, we actually move T such that T is more likely to be the reverse nearest neighbor of I . Hence, the MAP of query $\mathbb{Q}_{I \rightarrow T}$ is greatly improved. On the opposite, $\mathbb{Q}_{T \rightarrow I}$ decreases. From epoch 32 to epoch 60, the text latent features on the one hand move close to relevant image latent features slowly, on the other hand rebuild their intra-modal relationships. The latter is achieved by minimizing the reconstruction error, i.e., \mathcal{L}_r^T , to cap-

ture the regularities (semantics) of the original features. Therefore, both $\mathbb{Q}_{T \rightarrow T}$ and $\mathbb{Q}_{I \rightarrow T}$ grows gradually. Comparing Figure 11a and 11d, we can see the distance of relevant latent features in Figure 11d is larger than that in Figure 11a. The reason is that when β is larger, the objective function, i.e., Equation 8, pays more effort to minimize the reconstruction error \mathcal{L}_r^T . Consequently, less effort is paid to minimize the inter-modal distance \mathcal{L}_d . Hence, relevant inter-modal pairs cannot be moved closer. This effect is reflected as minor changes at epoch 31 in Figure 11f. Similarly, small changes happen between Figure 11d and 11e, which leads to minor changes from epoch 32 to 60 in terms of MAP in Figure 11f.

6.4 Evaluation of Model Effectiveness

6.4.1 NUS-WIDE dataset

We first examine the mean average precision (MAP) of our method compared using Euclidean distance against the real-valued features. Let L be the dimension of the latent space. Our MSAE is configured with 3 layers, where the image features are mapped from 500 dimensions to 128, and finally to L . Similarly, the dimension of text features are reduced from 1000 \rightarrow 128 $\rightarrow L$ by the text SAE. α and β are set to 0 and 0.01 respectively according to Section 6.3. We test L with values 16, 24 and 32. The results compared with other methods are reported in Table 2. Our MSAE achieves the best performance for all the four search tasks. It demonstrates an average improvement of 17%, 27%, 21%, and 26% for $\mathbb{Q}_{I \rightarrow I}$, $\mathbb{Q}_{T \rightarrow T}$, $\mathbb{Q}_{I \rightarrow T}$, and $\mathbb{Q}_{T \rightarrow I}$ respectively. CVH and CMSSH prefer smaller L in queries $\mathbb{Q}_{I \rightarrow T}$ and $\mathbb{Q}_{T \rightarrow I}$. The reason is that it needs to train far more parameters in higher dimensions and the learned models will be farther from the optimal solutions. Our method is less sensitive to the value of L . This is probably because with multiple layers, MSAE has stronger representation power and can better avoid local optimality by a good initialization from unsupervised pre-training, and thus is more robust under different L .

Figure 12 shows the precision-recall curves, and the recall-candidates ratio curves (used by [25, 26]) which show the change of recall when inspecting more results on the returned rank list. Due to the space limitation, we only show the results of $\mathbb{Q}_{T \rightarrow I}$ and $\mathbb{Q}_{I \rightarrow T}$. We achieve similar trends on results of $\mathbb{Q}_{T \rightarrow T}$ and $\mathbb{Q}_{I \rightarrow I}$. Our method shows the best accuracy except when recall is 0⁸, whose precision p implies that the nearest neighbor of the query appears in the $\frac{1}{p}$ -th returned result. This indicates that our method performs the best for general top- k similarity retrieval except $k=1$. For the measure of recall-candidates ratio, our method is always superior to other methods.

Besides real-valued features, we also conduct an experiment against binary latent features for which Hamming distance is used as the distance function. In our implementation, we choose Spectral Hashing [24] to convert real-valued latent feature vectors into binary codes. Other comparison algorithms use their own conversion mechanisms. The MAP scores are reported in Table 3. We can see that 1) MSAE still performs better than other methods. 2) The MAP

⁸Here, recall $r = \frac{1}{\#all\ relevant\ results} \approx 0$.

Table 3: Mean Average Precision on NUS-WIDE dataset (using Binary Latent Features)

Task		$\mathbb{Q}_{I \rightarrow I}$				$\mathbb{Q}_{T \rightarrow T}$				$\mathbb{Q}_{I \rightarrow T}$				$\mathbb{Q}_{T \rightarrow I}$			
Algorithm		LCMH	CMSSH	CVH	MSAE												
Dimension of Latent Space L	16	0.353	0.357	0.352	0.376	0.387	0.391	0.379	0.397	0.328	0.339	0.359	0.364	0.325	0.346	0.359	0.392
	24	0.347	0.358	0.346	0.368	0.392	0.396	0.372	0.412	0.333	0.346	0.353	0.371	0.324	0.352	0.353	0.380
	32	0.345	0.358	0.343	0.359	0.395	0.397	0.365	0.434	0.320	0.340	0.348	0.373	0.318	0.347	0.348	0.372

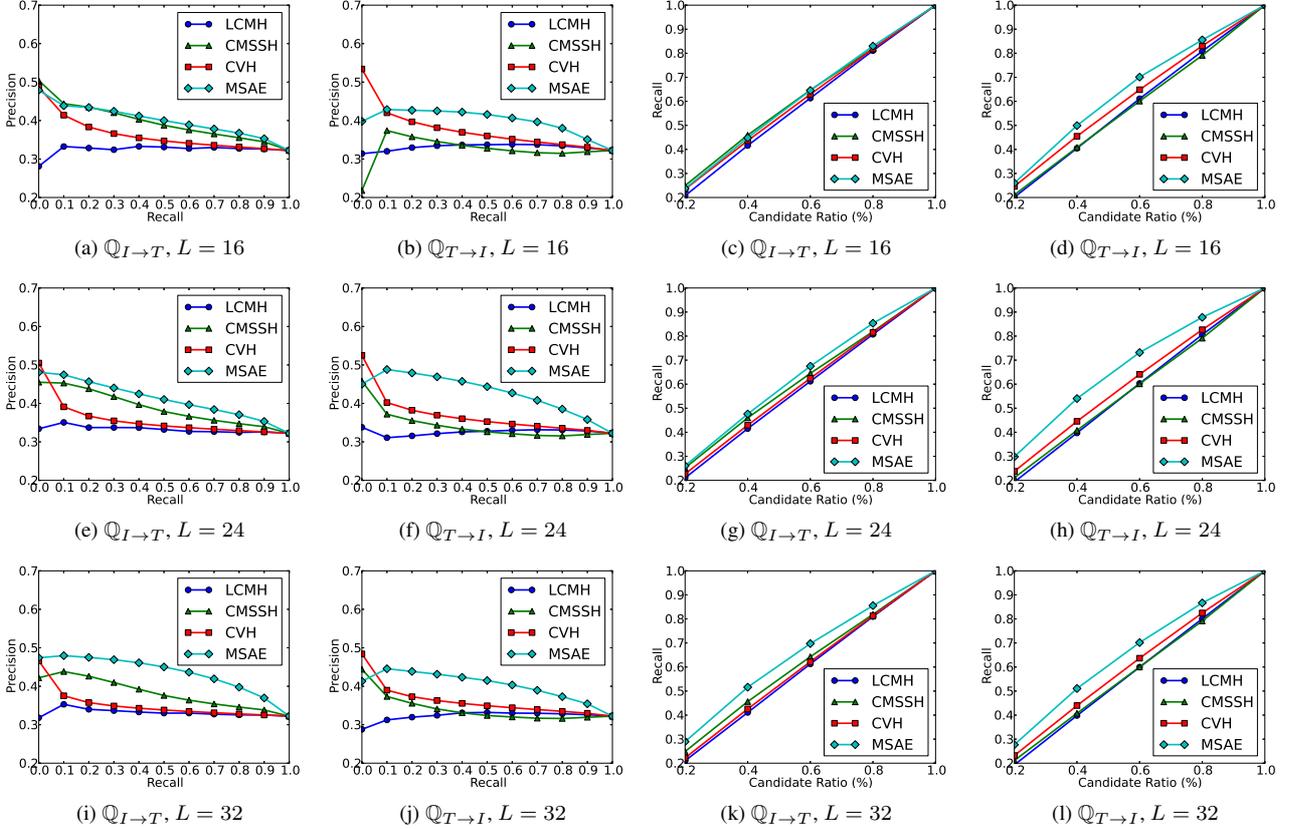


Figure 12: Precision-Recall and Recall-Candidates Ratio on NUS-WIDE dataset

scores using Hamming distance is not as good as Euclidean distance. This is caused by the information loss resulted from converting real-valued features into binary features.

6.4.2 Wiki Dataset

We conduct similar evaluations on **Wiki** dataset as on NUS-WIDE. For MSAE with latent feature of dimension L , the structure of its image SAE is $128 \rightarrow 128 \rightarrow L$, and the structure of its text SAE is $1000 \rightarrow 128 \rightarrow L$. Similar to the setting on NUS-WIDE, α is set to 0 due to the low quality of image features, and β is set to 0.01 to make \mathcal{L}_r^T and \mathcal{L}_d within the same scale.

The performance is report in Table 4. The MAPs on **Wiki** dataset are much smaller than those on NUS-WIDE except for $\mathbb{Q}_{T \rightarrow T}$. This is because the images of **Wiki** are of much lower quality. It contains only 2,000 images that are highly diversified, making it difficult to capture the semantic relationships between images and text. Query task $\mathbb{Q}_{T \rightarrow T}$ is not affected as Wikipedia’s featured articles are well edited and rich in text information. In general, our method achieves an average improvement of 8.1%, 30.4%, 32.8%, 26.8% for $\mathbb{Q}_{I \rightarrow I}$, $\mathbb{Q}_{T \rightarrow T}$, $\mathbb{Q}_{I \rightarrow T}$, and $\mathbb{Q}_{T \rightarrow I}$ respectively. We do not plot the precision-recall curves and recall-candidates ratio curves due to space limitation. Generally, these curves show

similar trends to those of NUS-WIDE except that the precisions are smaller due to low-quality images.

6.4.3 Flickr1M Dataset

We configure a 4-layer image SAE for this dataset as $3857 \rightarrow 1000 \rightarrow 128 \rightarrow L$, and the text SAE is configured as $2000 \rightarrow 1000 \rightarrow 128 \rightarrow L$. Because the image feature of this dataset consists of both local and global feature, its quality is better. In fact, the image latent feature performs equally well for intra-modal search as the text latent feature. Hence, we set both α and β to 0.01.

We compare the MAP of MSAE and CVH in Table 5. MSAE outperforms CVH in most of the search tasks. The results of LCMH and CMSSH cannot be reported as both methods run out of memory in the training stage.

6.5 Evaluation of Training Overhead

We use Flickr1M to evaluate the training time and memory consumption and report the results in Figure 13. The training cost of LCMH and CMSSH are not reported because they run out of memory on this dataset. We can see that the training time of MSAE

Table 4: Mean Average Precision on **Wiki** dataset

Task		$Q_{I \rightarrow I}$				$Q_{T \rightarrow T}$				$Q_{I \rightarrow T}$				$Q_{T \rightarrow I}$			
Algorithm		LCMH	CMSSH	CVH	MSAE												
Dimension of Latent Space L	16	0.146	0.148	0.147	0.162	0.359	0.318	0.153	0.462	0.133	0.138	0.126	0.182	0.117	0.140	0.122	0.179
	24	0.149	0.151	0.150	0.161	0.345	0.320	0.151	0.437	0.129	0.135	0.123	0.176	0.124	0.138	0.123	0.168
	32	0.147	0.149	0.148	0.162	0.333	0.312	0.152	0.453	0.137	0.133	0.128	0.187	0.119	0.137	0.123	0.179

Table 5: Mean Average Precision on **Flickr1M** dataset

Task		$Q_{I \rightarrow I}$		$Q_{T \rightarrow T}$		$Q_{I \rightarrow T}$		$Q_{T \rightarrow I}$	
Algorithm		CVH	MSAE	CVH	MSAE	CVH	MSAE	CVH	MSAE
Dimension of Latent Space L	16	0.622	0.621	0.610	0.624	0.610	0.632	0.616	0.608
	24	0.616	0.619	0.604	0.629	0.605	0.628	0.612	0.612
	32	0.603	0.622	0.587	0.630	0.588	0.632	0.598	0.614

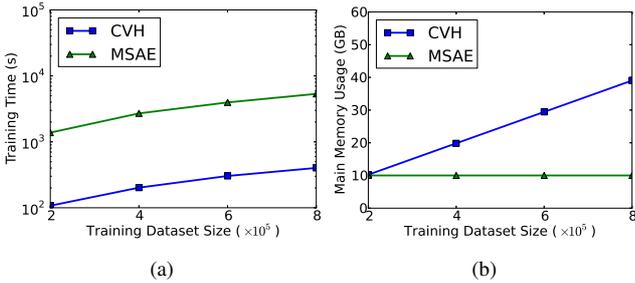


Figure 13: Training Time and Memory Consumption

and CVH increases linearly with respect to the size of the training dataset. Due to the stacked structure and multiple iterations of passing the dataset, MSAE is not as efficient as CVH. Roughly, the overhead is the number of training iterations times the height of MSAE. Possible solutions for accelerating the MSAE training include parallelizing the computation of image SAE and text SAE or adopting Distributed deep learning [3]. We leave this as our future work.

Figure 13b shows the memory usage of the training process. Given a training dataset, MSAE splits them into mini-batches and conducts the training batch by batch (see Algorithm 2). It stores the model parameters and one mini-batch in memory, both of which are independent of the training dataset size. Hence, the memory usage stays constant when the size of the training dataset increases. In fact, the minimum memory usage for MSAE is smaller than 10GB. We allocate more space to load multiple mini-batches into memory to save disk reading cost. For CVH, it has to load all training data into memory for matrix operations. Therefore, the memory usage increases with respect to the size of the training dataset.

6.6 Evaluation of Query Processing Efficiency

Finally, we compare the efficiency of query processing using binary latent features and real-valued latent features. Notice that all methods (i.e., MSAE, CVH, CMSSH and LCMH) perform similarly in query processing after mapping the original data into latent features of same dimensions. Data from the **Flickr1M** training dataset is mapped into a 32 dimensional latent space to form a large dataset for searching. To speed up the query processing of real-valued latent features, we create an index (i.e., VA-File [23]) for each modality. For binary latent features, we do not create any indexes, because linear scan is fast enough as shown in Figure 14. It shows the time (averaged over 100 random queries) of searching 50 nearest neighbors against datasets represented using binary latent features (based on Euclidean distance) and real-valued features (based on Hamming distance) respectively. We can see that

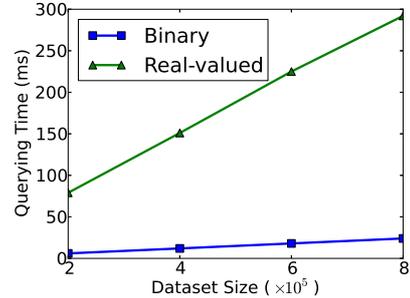


Figure 14: Querying Time Comparison Using Real-valued and Binary Latent Features

the querying time increases linearly with respect to the dataset size for both binary and real-valued latent features. But, the searching against binary latent features is 10 \times faster than that against real-valued latent features. This is because the computation of Hamming distance is more efficient than that of Euclidean distance. By taking into account the results from effectiveness evaluations, we can see that there is a trade-off between efficiency and effectiveness in feature representation. The binary encoding greatly improves the efficiency in the expense of accuracy degradation.

7. CONCLUSION

In this paper, we have proposed a new mapping mechanism for multi-modal retrieval based on the stacked auto-encoders (SAE). Our mapping mechanism, called multi-modal stacked auto-encoders (MSAE), learns a set of SAEs (one for each modality) to map the high-dimensional features of different media types (i.e., modalities) into a common low-dimensional latent space so that metric distance measures can be applied efficiently. By considering both the intra-modal semantics and the inter-modal semantics in the learning objective function, we learn a set of effective SAEs for feature mapping. Compared to existing methods which usually require a substantial amount of prior knowledge about the training data, our method requires little prior knowledge. Experiment results confirmed the improvements of our method over previous works in search accuracy.

8. REFERENCES

- [1] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR*, pages 3594–3601, 2010.
- [2] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., July 8-10, 2009.
- [3] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, K. Yang, and A. Y. Ng. Large scale distributed deep networks. In *NIPS*, pages 1232–1240, 2012.

- [4] R. Goroshin and Y. LeCun. Saturating auto-encoder. *CoRR*, abs/1301.3577, 2013.
- [5] G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Technical report, 2010.
- [6] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces. *ACM Trans. Database Syst.*, 28(4):517–580, 2003.
- [7] M. J. Huiskes and M. S. Lew. The mir flickr retrieval evaluation. In *Multimedia Information Retrieval*, pages 39–43, 2008.
- [8] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
- [9] S. Kumar and R. Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*, pages 1360–1365, 2011.
- [10] Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient BackProp. In G. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, chapter 2, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, Mar. 1998.
- [11] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.
- [12] X. Lu, F. Wu, S. Tang, Z. Zhang, X. He, and Y. Zhuang. A low rank structural large margin method for cross-modal ranking. In *SIGIR*, pages 433–442, 2013.
- [13] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, pages 151–175. Cambridge University Press, 2008.
- [14] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *ICML*, pages 689–696, 2011.
- [15] N. Rasiwasia, J. C. Pereira, E. Coviello, G. Doyle, G. R. G. Lanckriet, R. Levy, and N. Vasconcelos. A new approach to cross-modal multimedia retrieval. In *ACM Multimedia*, pages 251–260, 2010.
- [16] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, pages 833–840, 2011.
- [17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(Oct):533–536+, 1986.
- [18] R. Salakhutdinov and G. E. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.
- [19] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, pages 151–161, 2011.
- [20] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *SIGMOD Conference*, pages 785–796, 2013.
- [21] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, pages 2231–2239, 2012.
- [22] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.
- [23] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.
- [24] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.
- [25] Y. Zhen and D.-Y. Yeung. A probabilistic model for multimodal hash function learning. In *KDD*, pages 940–948, 2012.
- [26] X. Zhu, Z. Huang, H. T. Shen, and X. Zhao. Linear cross-modal hashing for efficient multimodal search. *MM*, 2013.
- [27] Y. Zhuang, Y. Yang, and F. Wu. Mining semantic correlation of heterogeneous multimedia data for cross-media retrieval. *IEEE Transactions on Multimedia*, 10(2):221–229, 2008.

APPENDIX

In this section we give the details of parameter updating, i.e., the adapted back-propagation procedure **bProp** in Algorithm 1. We omit the details for Algorithm 2, since they are similar to those in Algorithm 1. All following equations are in matrix form, and can be verified element-wisely. Parameters θ are updated according to the Stochastic Gradient Descent, i.e.,

$$\theta = \theta - \gamma * \frac{\partial \mathcal{L}}{\partial \theta} \quad (9)$$

where γ is a hyper-parameter, called learning rate. Specifically, to calculate the partial derivative of the objective function \mathcal{L} w.r.t. the weight matrix W and bias b , the partial derivative w.r.t. the activation Z of each layer is calculated firstly. For layer-wise training, i.e., **trainNN** with $h = 1$, the partial derivative is,

$$\begin{aligned} \frac{\partial \mathcal{L}(B^0)}{\partial Z^{2h}} &\stackrel{Eq\ 7}{=} \frac{\partial \mathcal{L}_r(B^0, B^{2h})}{\partial Z^{2h}} (\mathcal{L}_r \in \{\mathcal{L}_r^I, \mathcal{L}_r^T\}) \\ &= \begin{cases} (B^{2h} - B^0) * \frac{\partial s(Z^{2h})}{\partial Z^{2h}}, & \text{otherwise} \quad (10a) \\ B^{2h} - B^0, & \text{bottom auto-encoder} \quad (10b) \end{cases} \end{aligned}$$

Equation 10a is for auto-encoders in the upper layers as shown in Figure 2. We use the Sigmoid function for the encode activation function $s_e()$ and decode activation function $s_d()$, uniformly denoted as $s()$. The partial derivative of the Sigmoid function is $s(Z) * (1 - s(Z))$, where $*$ stands for element-wise multiplication. For the bottom auto-encoder, it has modal specific activation function for the reconstruction layer and error function, thus has different partial derivatives, as shown by Equation 10b. The bottom auto-encoders for the image modality and the text modality share the same partial derivative by coincidence. For fine-tuning of SAE, i.e., **trainNN** with $h > 1$, the activation function of the last reconstruction layer and error function are the same to those of the bottom auto-encoder in layer-wise training. Hence it has the same partial derivative, i.e., Equation 10b.

With the above partial derivative, denoted as δ^{2h} , we can calculate the partial derivative for W and b in the i -th layer of Figure 7 as,

$$\begin{aligned} \frac{\partial \mathcal{L}(B^0)}{\partial W^i} &\stackrel{Eq\ 7}{=} \frac{\partial \mathcal{L}_r(B^0, B^{2h})}{\partial W^i} + W^i * \xi \\ &= B^{i-1T} \delta^i + W^i * \xi \\ \frac{\partial \mathcal{L}(B^0)}{\partial b^i} &\stackrel{Eq\ 7}{=} \frac{\partial \mathcal{L}_r(B^0, B^{2h})}{\partial b^i} = \sum_j \delta_j^i \end{aligned}$$

To update parameters in the $i - 1$ th layer, we have to calculate the δ^{i-1} firstly,

$$\delta^{i-1} = \frac{\partial \mathcal{L}(B^0)}{\partial Z^{i-1}} = \delta^i W^{iT} * \frac{\partial s(Z^{i-1})}{\partial Z^{i-1}}$$