VLDBJ manuscript No.
(will be inserted by the editor)

Effective Deep Learning Based Multi-Modal Retrieval

Wei Wang, Xiaoyan Yang, Beng Chin Ooi, Dongxiang Zhang, Yueting Zhuang

the date of receipt and acceptance should be inserted later

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 for relevant reviews and trailers. The mainstream solution to the problem is to learn a set of mapping functions that project data from different modalities into a common metric space where the retrieval is then conducted. Hence, how to design effective mapping functions plays an important role in improving search quality. In this paper, we exploit deep learning techniques to learn effective mapping functions. In our framework, we propose a general learning objective that effectively captures both intra-modal and inter-modal semantic relationships of data from heterogeneous sources. Given the general objective, we propose two learning algorithms to realize it: (1) an unsupervised approach based on stacked auto-encoders (SAEs) which requires minimum prior knowledge on the training data, and (2) a supervised approach based on deep convolutional neural network (DCNN) and neural language model (NLM) which have shown great success in learning features for vision and text data respectively. Our training algorithms are memory efficient with respect to data volume. Large training dataset is split into mini-batches and the mapping functions are continuously adjusted for each batch. Experiments on three real datasets illustrate that our proposed methods achieve significant improvement in search accuracy over the state-of-the-art solutions.

Wei Wang
School of Computing, National University of Singapore, Singapore.
E-mail: wangwei@comp.nus.edu.sg

Xiaoyan Yang
Advanced Digital Sciences Center, Illinois at Singapore Pte, Singapore.
E-mail: xiaoyan.yang@adsc.com.sg

Beng Chin Ooi
School of Computing, National University of Singapore, Singapore.
E-mail: ooibc@comp.nus.edu.sg

Dongxiang Zhang
School of Computing, National University of Singapore, Singapore.
E-mail: zhangdo@comp.nus.edu.sg

Yueting Zhuang
College of Computer Science and Technology, Zhejiang University, Hangzhou, China
E-mail: yzhuang@zju.edu.cn

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 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 day[1], while Facebook reported that around 300 million photos were created each day[2]. These data, together with other domain specific data, such as medical data, surveillance and sensory data, are big data that can be exploited for insights and contextual observations. However, effective retrieval of such huge amounts of media from heterogeneous sources remains a big challenge.

In this paper, we exploit deep learning techniques, which have been successfully applied in processing media data [33, 29, 3], to solve the problem of large-scale information retrieval from multiple modalities. Each modality represents one type of media such as text, image or video. Depending on the heterogeneity of data sources, we have two types of searches:

[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/

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 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 [44, 4, 43, 36, 30, 21, 42, 25]. These works follow the same query processing strategy, which consists of two major steps. First, a set of mapping functions are learned to project data from different modalities into a common latent space. Second, a multi-dimensional index for each modality in the common metric space is built for efficient similarity retrieval. Since the second step is a classic kNN problem and has been extensively studied [16, 40], we focus on the optimization of the first step and propose two types of novel mapping functions based on deep learning techniques.

We propose a general learning objective that effectively captures both intra-modal and inter-modal semantic relationships of data from heterogeneous sources. In particular, we differentiate modalities in terms of their representations' ability to capture semantic information and robustness when noisy data are involved. The modalities with better representations are assigned with higher weight for the sake of learning more effective mapping functions. Based on the objective function, we design an unsupervised algorithm using stacked auto-encoders (SAEs). SAE is a form of deep learning that has been widely used in many unsupervised feature learning and classification tasks [31, 38, 13, 34]. If the media are annotated with semantic labels, we can utilize such information to further improve the effectiveness by exploiting a deep convolutional neural network (DCNN) and neural language model (NLM) in a supervised manner. The label information can help learn robust mapping functions against noisy input data. DCNN and NLM have shown great success in learning image features [20, 10, 8] and text features [33, 28] respectively.

Compared with existing solutions for multi-modal retrieval, our approaches exhibit three major advantages. First, our mapping functions are non-linear, which are more expressive than the linear projections used in IMH [36] and CVH [21]. The deep structures of our models can capture more abstract concepts at higher layers, which is very useful in modeling categorical information of data for effective retrieval. Second, we require minimum prior knowledge in the training. Our unsupervised approach only needs relevant data pairs from different modalities as the training input. The supervised approach requires additional labels for the media objects. By contrast, MLBE [42] and IMH [36] require one similarity matrix of intra-modal data for each modality. LSCMR [25] uses training examples, each of which consists of a list of objects ranked according to their relevance (based on manual labels) to the first one. Third, our training process is memory efficient because we split the training dataset into mini-batches and iteratively load and train each mini-batch in memory. However, many existing works (e.g., CVH, 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.

The main contributions of this paper are:

- We propose a general learning objective for learning mapping functions to project data from different modalities into a common latent space for multi-modal retrieval. The learning objective differentiate modalities in terms of their input features' quality of capturing semantics.
- We realize the general learning objective by one unsupervised approach and one supervised approach based on deep learning techniques.
- 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 remainder of the paper is organized as follows. Problem statements and overview are provided in Section 2 and Section 3. After that, we describe the unsupervised and supervised approaches in Section 4 and Section 5 respectively. Query processing is described in Section 6. Then we dicuss related works in Section 7 and present our experimental study in Section 8. We conclude our paper in Section 9. This work in an extended version of [39] which proposed an unsupervised learning algorithm for multi-modal retrieval. A supervised learning algorithm is added in this work. Correspondingly, we add Section 3 to unify the learning objective of the two approaches. Section 5 and Section 8.3 are newly added for describing the supervised approach and its experimental results.

2 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 sample modalities to explain our idea, i.e., we assume that $\mathbb{D} = \mathbb{D}_I \bigcup \mathbb{D}_T$, even though our proposal is general enough to support other domains. An image (resp. a text document) is represented by a feature vector $x \in \mathbb{D}_I$ (resp. $y \in \mathbb{D}_T$). 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.

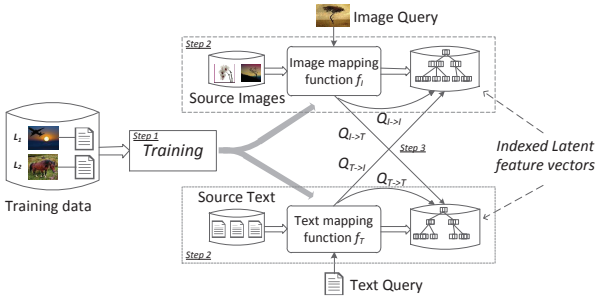


Fig. 1: Flowchart of multi-modal retrieval framework. Step 1 is offline model training that learns mapping functions. Step 2 is offline indexing that maps source objects into latent features and creates proper indexes. Step 3 is online multi-modal kNN query processing.

Definition 1 Common Latent Space Mapping
Given an image $x \in \mathbb{D}_I$ and a text document $y \in \mathbb{D}_T$, 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 $f_T(y)$ in the common latent space $\mathbb{Z}$, denoted by $dist_{\mathbb{Z}}(f_I(x), f_T(y))$, is small.

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. Once the mapping functions f_I and f_T have been determined, the multi-modal search can then be transformed into the classic kNN problem, defined as following:

Definition 2 Multi-Modal Search
Given a query object $Q \in \mathbb{D}_q$ and a target domain $\mathbb{D}_t$ ($q, t \in \{I, T\}$), find a set $O \subset \mathbb{D}_t$ with k objects such that $\forall o \in O$ and $o' \in \mathbb{D}_t/O, dist_{\mathbb{Z}}(f_q(Q), f_t(o')) \geq dist_{\mathbb{Z}}(f_q(Q), f_t(o))$.

Since both q and t have two choices, four types of queries can be derived, namely $\mathbb{Q}_{q \rightarrow t}$ and $q, t \in \{I, T\}$. For instance, $\mathbb{Q}_{I \rightarrow T}$ searches relevant text in $\mathbb{D}_T$ given an image from $\mathbb{D}_I$. By mapping objects from different high-dimensional feature spaces into a low-dimensional latent space, queries can be efficiently processed using existing multi-dimensional indexes [16, 40]. 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 of mapping functions is measured by the accuracy of multi-modal retrieval using latent features.

3 Overview of Multi-modal Retrieval

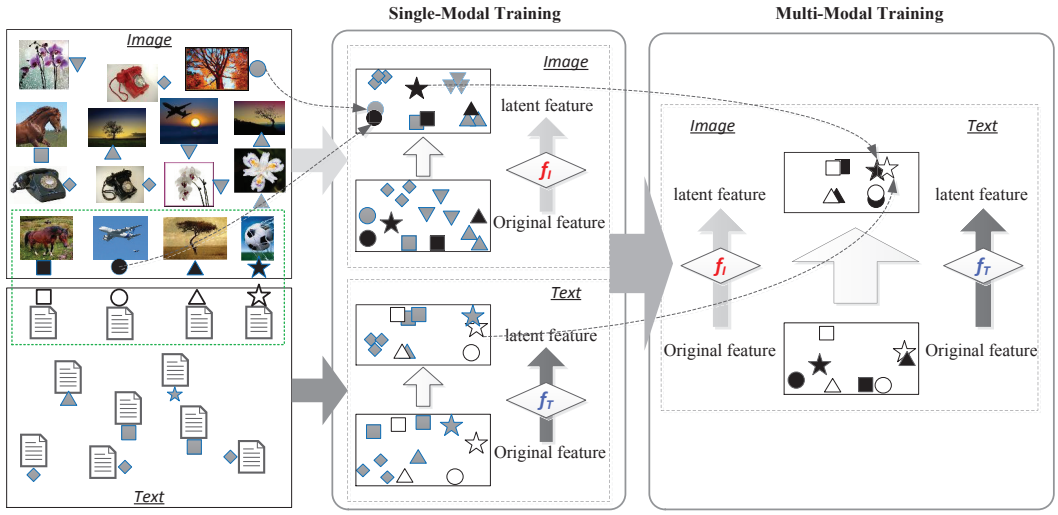
The flowchart of our multi-modal retrieval framework is illustrated in Figure 1. It consists of three main steps: 1) offline model training 2) offline indexing 3) online kNN query processing. In step 1, relevant image-text pairs are used as input data for our training model. For example, image-text pairs can be collected from Flickr where the text features are extracted from tags and descriptions for images. If they are associated with additional semantic labels (e.g., categories), we use a supervised training model. Otherwise, an unsupervised training model is used. After step 1, we can obtain a mapping function $f_m : \mathbb{D}_m \rightarrow \mathbb{Z}$ for each modality $m \in \{I, T\}$. In step 2, objects from different modalities are first mapped into the common space $\mathbb{Z}$ by function f_m. With such unified representation, the latent features from the same modality are then inserted into a high dimensional index for kNN query processing. When a query $Q \in \mathbb{D}_m$ comes, it is first mapped into $\mathbb{Z}$ using its modal-specific mapping function f_m. Based on the query type, k nearest neighbors are retrieved from the index built for the target modality and returned to the user. For example, image index is used for queries of type $\mathbb{Q}_{I \rightarrow I}$ and $\mathbb{Q}_{T \rightarrow I}$ against the image database.

General learning objective A good objective function plays a crucial role in learning effective mapping functions. In our multi-modal search framework, we design a general learning objective function $\mathcal{L}$. By taking into account the image and text modalities, our objective function is defined as follows:

$$\mathcal{L} = \beta_I \mathcal{L}_I + \beta_T \mathcal{L}_T + \mathcal{L}_{I,T} + \xi(\theta) \tag{1}$$

where $\mathcal{L}_m$, $m \in \{I, T\}$ is called the intra-modal loss to reflect how well the intra-modal semantics are captured by the latent features. The smaller the loss, the more effective the training model is. $\mathcal{L}_{I,T}$ is called inter-modal loss which is designed to capture inter-modal semantics. The last term is used as regularization to prevent over-fitting [14] (L_2 Norm is used in our experiment). θ denotes all parameters involved in the mapping functions. β_m, $m \in \{I, T\}$ denotes the weight of the loss for modality m in the objective function. We observe in our training process that assigning different weights to different modalities according to the nature of its data offers better performance than treating them equally. For the modality with lower quality input feature (due to noisy data or poor data representation), we assign smaller weight for its intra-modal loss in the objective function. The *intuition* of setting β_I and β_T in this way is that, by relaxing the constraints on intra-modal loss, we enforce the inter-modal constraints. Consequently, the intra-modal semantics of the modality with lower quality input feature can be preserved or even enhanced through their inter-modal relationships with high-quality modalities. Details of setting β_I and β_T will be discussed in Section 4.3 and Section 5.4.

Training Training is to find the optimal parameters involved in the mapping functions that minimizes $\mathcal{L}$. Two types of mapping functions are proposed in this paper. One is trained



(a)Input (b) Training Stage I (c) Training Stage II

Fig. 2: Flowchart of training. Relevant images (or text) are associated with the same shape (e.g., ■). In single-modal training, objects of same shape and modality are moving close to each other. In multi-modal training, objects of same shape from all modalities are moving close to each other.

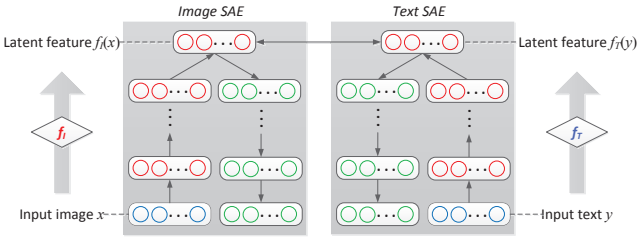


Fig. 3: Model of MSAE, which consists of one SAE for each modality. The trained SAE maps input data into latent features.

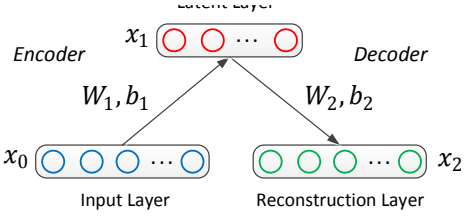


Fig. 4: Auto-Encoder

by an unsupervised algorithm, which uses simple image-text pairs for training. No other prior knowledge is required. The other one is trained by a supervised algorithm which exploits additional label information to learn robust mapping functions against noisy training data. For both mapping functions, we design a two-stage training procedure to find the optimal parameters. A complete training process is illustrated in Figure 2. In *stage slowromancapi@*, one mapping function is trained independently for each modality with the objective to map similar features in one modality close to each other in the latent space. This training stage serves as the pre-training of stage slowromancapii@ by providing a good initialization for the parameters. In *stage slowromancapii@*, we jointly optimize Equation 1 to capture both intra-modal semantics and inter-modal semantics. The learned mapping functions project semantically relevant objects close to each other in the latent space as shown in the figure.

4 Unsupervised Approach – MSAE

In this section, we propose an unsupervised learning algorithm called **MSAE** (Multi-modal Stacked Auto-Encoders) to learn the mapping function f_I and f_T. The model is shown in Figure 3. We first present the preliminary knowledge of auto-encoder and stacked auto-encoders. Based on stacked auto-encoders, we address how to define the terms $\mathcal{L}_I$, $\mathcal{L}_T$ and $\mathcal{L}_{I,T}$ in our general objective learning function in Equation 1.

4.1 Background: Auto-encoder & Stacked Auto-encoder

Auto-encoder Auto-encoder has been widely used in unsupervised feature learning and classification tasks [31, 38, 13, 34]. 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 4, the raw input feature $x_0 \in \mathcal{R}_{d_0}$ in the input layer is **encoded** into latent feature $x_1 \in \mathcal{R}^{d_1}$ via a deterministic mapping f_e:

$$x_1 = f_e(x_0) = s_e(W_1^T x_0 + b_1) \quad (2)$$

where s_e is the activation function of the encoder, $W_1 \in \mathbb{R}^{d_0 \times d_1}$ is a weight matrix and $b_1 \in \mathcal{R}^{d_1}$ is a bias vector. The latent feature x_1 is then **decoded** back to $x_2 \in \mathcal{R}^{d_0}$ via another mapping function f_d:

$$x_2 = f_d(x_1) = s_d(W_2^T x_1 + b_2) \tag{3}$$

Similarly, s_d is the activation function of the decoder with parameters $\{W_2, b_2\}$, $W_2 \in \mathbb{R}^{d_1 \times d_0}$, $b_2 \in \mathcal{R}^{d_0}$. Sigmoid function or Tanh function is typically used as the activation functions s_e and s_d. The parameters $\{W_1, W_2, b_1, b_2\}$ of the auto-encoder are learned with the objective of minimizing the difference (called reconstruction error) between the raw input x_0 and the reconstruction output x_2. Squared Euclidean distance, negative log likelihood and cross-entropy are often used to measure the reconstruction error. By minimizing the reconstruction error, we can use the latent feature to reconstruct the original input with minimum information loss. In this way, the latent feature preserves regularities (or semantics) of the input data.

Stacked Auto-encoder Stacked Auto-encoders (SAE) are constructed by stacking multiple (e.g., h) auto-encoders. The input feature vector x_0 is fed to the bottom auto-encoder. After training the bottom auto-encoder, the latent representation x_1 is propagated to the higher auto-encoder. The same procedure is repeated until all the auto-encoders are trained. The latent representation x_h from the top (i.e., h-th) auto-encoder, is the output of the stacked auto-encoders, which can be further fed into other applications, such as SVM for classification. The stacked auto-encoders can be fine tuned by minimizing the reconstruction error between the input feature x_0 and the reconstruction feature x_{2h} which is computed by forwarding the x_0 through all encoders and then through all decoders as shown in Figure 5. In this way, the output feature x_h can reconstruct the input feature with minimal information loss. In other words, x_h preserves regularities (or semantics) of the input data x_0.

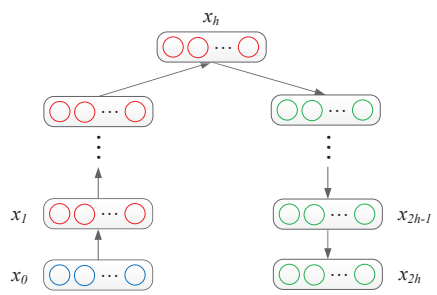
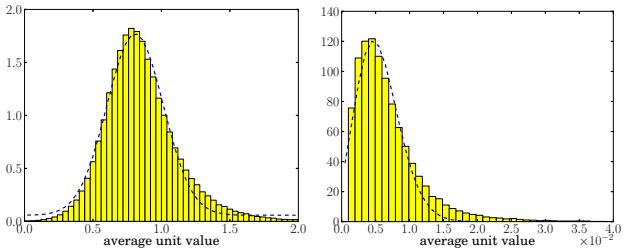


Fig. 5: Fine-tune Stacked Auto-Encoders



(a) (b)

Fig. 6: Distribution of image (6a) and text (6b) features extracted from NUS-WIDE training dataset (See Section 8). Each figure is generated by averaging the units for each feature vector, and then plot the histogram for all data.

4.2 Realization of Learning Objective in MSAE

4.2.1 $\mathcal{L}_I$ and $\mathcal{L}_T$: Modeling Intra-modal Semantics of Data

We extend stacked auto-encoders to model intra-modal losses involved in the general learning objective (Equation 1). Specifically, $\mathcal{L}_I$ and $\mathcal{L}_T$ are modeled as the reconstruction error for the image stacked auto-encoders and the text stacked auto-encoders respectively. Intuitively, if the two reconstruction errors are small, the latent features generated by the top auto-encoder would be able to reconstruct the original input well, and consequently, capture the regularities of the input data well. This implies that, with small reconstruction error, two objects from the same modality that are similar in the original space would also be close in the latent space. In this way, we are able to capture the intra-modal semantics of data by minimizing $\mathcal{L}_I$ and $\mathcal{L}_T$ respectively. But to use the stacked auto-encoders, we have to design the decoders of the bottom auto-encoders carefully to handle different input features.

The raw (input) feature of an image is a high dimensional real-valued vector (e.g., color histogram or bag-of-visual-words). In the encoder, each input image feature is mapped to a latent vector using Sigmoid function as the activation function s_e (Equation 2). However, in the decoder, the Sigmoid activation function, whose range is [0,1], performs poorly on reconstruction because the raw input unit (referring to one dimension) is not necessarily within [0,1]. To solve this issue, we follow Hinton [14] and model the raw input unit as a linear unit with independent Gaussian noise. As shown in Figure 6a, the average unit value of image feature typically follows Gaussian distribution. When the input data is normalized with zero mean and unit variance, the Gaussian noise term can be omitted. In this case, we can use an identity function for the activation function s_d in the bottom decoder. Let x_0 denote the input image feature vector, x_{2h} denote the feature vector reconstructed from the top latent feature x_h (h is the depth of the stacked auto-encoders).

Using Euclidean distance to measure the reconstruction error, we define $\mathcal{L}_I$ for x_0 as:

$$\mathcal{L}_I(x_0) = ||x_0 - x_{2h}||_2^2 \tag{4}$$

The raw (input) feature of text is a word count vector or tag occurrence vector [3]. We adopt the Rate Adapting Poisson model [32] for reconstruction because the histogram for the average value of text input unit generally follows Poisson distribution (Figure 6b). In this model, the activation function in the bottom decoder is

$$x_{2h} = s_d(z_{2h}) = l\frac{e^{z_{2h}}}{\sum_j e^{z_{2h_j}}} \tag{5}$$

where $l = \sum_j x_{0j}$ is the number of words in input text, and $z_{2h} = W_{2h}^T x_{2h-1} + b_{2h}$. The probability of a reconstruction unit x_{2h_i} being the same as the input unit x_{0_i} is:

$$p(x_{2h_i} = x_{0_i}) = Pois(x_{0_i}, x_{2h_i}) \tag{6}$$

where $Pois(n, \lambda) = \frac{e^{-\lambda}\lambda^n}{n!}$. Based on Equation 6, we define $\mathcal{L}_T$ using negative log likelihood:

$$\mathcal{L}_T(x_0) = -log \prod_i p(x_{2h_i} = x_{0_i}) \tag{7}$$

By minimizing $\mathcal{L}_T$, we require x_{2h} to be similar as x_0. In other words, the latent feature x_h is trained to reconstruct the input feature well, and thus preserves the regularities of the input data well.

4.2.2 $\mathcal{L}_{I,T}$: Modeling Inter-modal Semantics of Data

Given one relevant image-text pair (x_0, y_0), we forward them through the encoders of their stacked auto-encoders to generate latent feature vectors (x_h, y_h) (h is the height of the stacked auto-encoders). The inter-modal loss is then defined as,

$$\mathcal{L}_{I,T}(x_0, y_0) = dist(x_h, y_h) = ||x_h - y_h||_2^2 \tag{8}$$

By minimizing $\mathcal{L}_{I,T}$, we capture the inter-modal semantics of data. The intuition is quite straightforward: if two objects x_0 and y_0 are relevant, the distance between their latent features x_h and y_h shall be small.

4.3 Training

Following the training flow shown in Figure 2, in stage slowromancapi@ we train a stacked auto-encoder for the image modality and a stacked auto-encoders for the text modality separately. Back-Propagation [22] (see Appendix) is used to calculate the gradients of the objective loss, i.e., $\mathcal{L}_I$ or $\mathcal{L}_T$, w.r.t., the parameters. Then the parameters are updated according to mini-batch Stochastic Gradient Descent (SGD) (see Appendix), which averages the gradients contributed by a mini-batch of training records (images or text documents) and then adjusts the parameters. The learned image and text SAEs are fine-tuned in stage slowromancapii@ by Back-Propagation and mini-batch SGD with the objective to find the optimal parameters that minimize the learning objective (Equation 1). In our experiment, we observe that the training would be more stable if we alternatively adjust one SAE with the other SAE fixed.

Setting $\boldsymbol{\beta_I}$ & $\boldsymbol{\beta_T}$ β_I and β_T are the weights of the reconstruction error of image and text SAEs respectively in the objective function (Equation 1). As mentioned in Section 3, they are set based on the quality of each modality's raw (input) feature. We use an example to illustrate the intuition. Consider a relevant object pair (x_0, y_0) from modality x and y. Assume x's feature is of low quality in capturing semantics (e.g., due to noise) while y's feature is of high quality. If x_h and y_h are the latent features generated by minimizing the reconstruction error, then y_h can preserve the semantics well while x_h is not as meaningful due to the low quality of x_0. To solve this problem, we combine the inter-modal distance between x_h and y_h in the learning objective function and assign smaller weight to the reconstruction error of x_0. This is the same as increasing the weight of the inter-modal distance from x_h to y_h. As a result, the training algorithm will move x_h towards y_h to make their distance smaller. In this way, the semantics of low quality x_h could be enhanced by the high quality feature y_h.

In the experiment, we evaluate the quality of each modality's raw feature on a validation dataset by performing intra-modal search against the latent features learned in single-modal training. Modality with worse search performance is assigned a smaller weight. Notice that, because the dimensions of the latent space and the original space are usually of different orders of magnitude, the scale of $\mathcal{L}_I$, $\mathcal{L}_T$ and $\mathcal{L}_{I,T}$ are different. In the experiment, we also scale β_I and β_T to make the losses comparable, i.e., within an order of magnitude.

5 Supervised Approach–MDNN

In this section, we propose a supervised learning algorithm called **MDNN** (Multi-modal Deep Neural Network) based on a deep convolutional neural network (DCNN) model and a neural language model (NLM) to learn mapping functions for the image modality and the text modality respectively. The model is shown in Figure 7. First, we provide some background on DCNN and NLM. Second, we extend one DCNN [20] and one NLM [28] to model intra-modal losses involved in the general learning objective (Equation 1). Third,

[3] The binary value for each dimension indicates whether the corresponding tag appears or not.

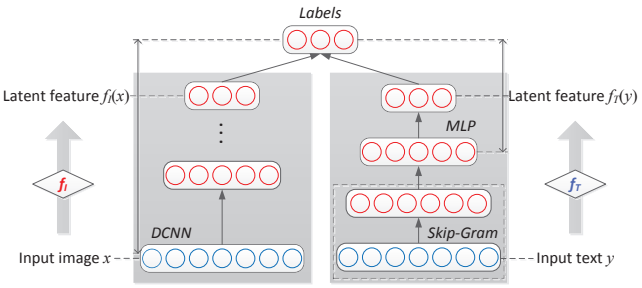


Fig. 7: Model of MDNN, which consists of one DCNN for image modality, and one Skip-Gram + MLP for text modality. The trained DCNN (or Skip-Gram + MLP) maps input data into latent features.

the inter-modal loss is specified and combined with the intra-modal losses to realize the general learning objective. Finally, we describe the training details.

5.1 Background: Deep Convolutional Neural Network & Neural Language Model

Deep Convolutional Neural Network (DCNN) DCNN has shown great success in computer vision tasks [8, 10] since the first DCNN (called AlexNet) was proposed by Alex [20]. It has specialized connectivity structure, which usually consists of multiple convolutional layers followed by fully connected layers. These layers form stacked, multiple-staged feature extractors, with higher layers generating more abstract features from lower ones. On top of the feature extractor layers, there is a classification layer. Please refer to [20] for a more comprehensive review of DCNN.

The input to DCNN is raw image pixels such as an RGB vector, which is forwarded through all feature extractor layers to generate a feature vector that is a high-level abstraction of the input data. The training data of DCNN consists of image-label pairs. Let x denote the image raw feature and $f_I(x)$ the feature vector extracted from DCNN. t is the binary label vector of x. If x is associated with the i-th label l_i, t_i is set to 1 and all other elements are set to 0. $f_I(x)$ is forwarded to the classification layer to predict the final output $p(x)$, where $p_i(x)$ is the probability of x being labelled with l_i. Given x and $f_I(x)$, $p_i(x)$ is defined as:

$$p_i(x) = \frac{e^{f_I(x)_i}}{\sum_j e^{f_I(x)_j}} \tag{9}$$

which is a softmax function. Based on Equation 9, we define the prediction error, or *softmax loss* as the negative log likelihood:

$$\mathcal{L}_I(x,t) = -\sum_i t_i \log p_i(x) \tag{10}$$

Neural Language Model (NLM) NLMs, first introduced in [2], learn a dense feature vector for each word or phrase, called a distributed representation or a *word embedding*. Among them, the Skip-Gram model (SGM) [28] proposed by Mikolov *et al*. is the state-of-the-art. Given two words a and b that co-occur, SGM models the conditional probability $p(a|b)$ using softmax:

$$p(a|b) = \frac{e^{v_a \cdot v_b}}{\sum_{\tilde{a}} e^{v_{\tilde{a}} \cdot v_b}} \tag{11}$$

where v_a and v_b are vector representations of words a and b respectively. The denominator $\sum_{\tilde{a}} e^{v_{\tilde{a}} \cdot v_b}$ is expensive to calculate, where $\tilde{a}$ is any word in the vocabulary. Thus, approximations were proposed to estimate it [28]. Given a corpus of sentences, SGM is trained to learn vector representations v by maximizing Equation 11 over all co-occurring pairs.

Second, the learned dense vectors can be used to construct a dense vector for one sentence or document (e.g., by averaging), or to calculate the similarity of two words, e.g., using the cosine similarity function.

5.2 Realization of Learning Objective in MDNN

5.2.1 Modeling Intra-modal Semantics of Data

Having witnessed the outstanding performance of DCNNs in learning features for visual data [8, 10], and NLMs in learning features for text data [33], we extend one instance of DCNN – AlexNet [20] and one instance of NLM – Skip-Gram model (SGM) [28] to model the intra-modal semantics of images and text respectively.

Image We employ AlexNet to serve as the mapping function $f_I()$ for image modality. An image x is represented by an RGB vector. The feature vector $f_I(x)$ learned by AlexNet is used to predict the associated labels of x. However, the objective of the original AlexNet is for to predict single label of an image while in our case images are annotated with multiple labels. We thus follow [11] to extend the softmax loss (Equation 10) to handle multiple labels as follows

$$\mathcal{L}_I(x,t) = -\frac{1}{\sum_i t_i} \sum_i t_i \log p_i(x) \tag{12}$$

where $p_i(x)$ is defined in Equation 9. Different from SAE, which models reconstruction error to preserve intra-modal semantics, the extended AlexNet tries to minimize the prediction error $\mathcal{L}_I()$ shown in Equation 12. By minimizing prediction error, we require the learned high-level feature vectors $f_I(x)$ to be discriminative in predicting labels. Images with similar labels shall have similar feature vectors. In this way, the intra-modal semantics are preserved.

Text We extend SGM to learn the mapping function $f_T()$ for text modality. Due to the noisy nature of text (e.g., tags)

associated with images [23], directly training the SGM over the tags would carry noise into the learned features. However, labels associated with images are carefully annotated and more accurate. Hence, we extend the SGM to integrate label information so as to learn robust features against noisy text (tags). The main idea is, we first pre-train a SGM [28], treating all tags associated with one image as an input sentence. After training, we obtain one word embedding for each tag. By averaging word embeddings of all tags of one image, we create one text feature vector for those tags. Second, we build a MultiLayer Perceptron (MLP) with two hidden layers on top of the SGM. The text feature vectors are fed into the MLP to predict image labels. Let y denote the input text (e.g., a set of image tags), $\tilde{y}$ denote the averaged word embedding generated by SGM for tags in y. MLP together with SGM serves as the mapping function $f_T()$ for the text modality,

$$f_T(y) = W_2 \cdot s(W_1\tilde{y} + b_1) + b_2 \tag{13}$$

$$s(v) = max(0, v) \tag{14}$$

where W_1 and W_2 are weight matrices, b_1 and b_2 are bias vectors, and $s()$ is the ReLU activation function [20][4]. The loss function of MLP is similar to that of the extended AlexNet for image label prediction:

$$\mathcal{L}_T(y,t) = -\frac{1}{\sum_i t_i}\sum_i \log q_i(y) \tag{15}$$

$$q_i(y) = \frac{e^{f_T(y)_i}}{\sum_j e^{f_T(y)_j}} \tag{16}$$

We require the learned text latent features $f_T(y)$ to be discriminative for predicting labels. In this way, we model the intra-modal semantics for the text modality [5].

5.3 Modeling Inter-modal Semantics

After extending the AlexNet and Skip-Gram model to preserve the intra-modal semantics for images and text respectively. The inter-modal semantics between image and text however is not considered, we jointly learn the latent features for image and text to preserve the inter-modal semantics. We follow the general learning objective in Equation 1 and realize $\mathcal{L}_I$ and $\mathcal{L}_T$ using Equation 12 and 15 respectively. Euclidean distance is used to measure the difference of the latent features for an image-text pair. By minimizing the distance of latent features for an image-text pair, we require their latent features to be closer in the latent space. In this way, the inter-modal semantics are preserved. The joint objective loss is (with the regularization term omitted),

$$\mathcal{L}(x,y,t) = \beta_I\mathcal{L}_I(x,t) + \beta_T\mathcal{L}_T(y,t) + ||f_I(x) - f_T(y)||_2^2 \tag{17}$$

5.4 Training

Similar to the training of MSAE, the training of MDNN takes two steps. The first step trains the extended AlexNet and the extended NLM (i.e., MLP+Skip-Gram) separately[6]. The learned parameters are used to initialize the joint model. All training is conducted by Back-Propagation using mini-batch SGD (see Appendix) to minimize the objective loss.

Setting $\boldsymbol{\beta_I}$ & $\boldsymbol{\beta_T}$ In the unsupervised training, we assign larger β_I to make the training prone to preserve the intra-modal semantics of images if the input image feature is of higher quality than the text input feature, vice versa. Here for supervised training, since the intra-modal semantics are preserved based on reliable labels, we do not distinguish the weight for image part and text part in the joint training. Hence β_I and β_T are set to the same value. In our experiment, to make the three losses within one order of magnitude, we set $\beta_I = \beta_T = 1$ and multiply the Euclidean distance with a weight of 0.01.

6 Query Processing

After the unsupervised (or supervised) training, each modality has a mapping function whose parameters are already well learned. Given a set of heterogeneous data sources, high-dimensional raw features (e.g., bag-of-visual-words or RGB feature for images) are extracted from each source and mapped into a common latent space using the learned mapping functions. If the mapping functions are learned by MSAE, then we use the image (resp. text) SAE to project image (resp. text) input features into the latent space. If the mapping functions are learned by MDNN, then we use the extended DCNN (resp. extended NLM) to map the image (resp. text) input feature into the common latent space.

After the mapping, we create VA-Files [40] over the latent features (one per modality). VA-File is a classic index that can overcome the curse of dimensionality when answering nearest neighbor queries. It encodes each data point into a bitmap and the whole bitmap file is loaded into memory for efficient scanning and filtering. Only a small number of real data points will be loaded into memory for verification. Given a query input, we check its media type and

[4] We tried both the Sigmoid function and ReLU activation function for $s()$. ReLU offers better performance.

[5] Notice that in our model, we fix the word vectors learned by SGM. It can also be fine-tuned by integrating the objective of SGM (Equation 11) into Equation 15.

[6] In our experiment, we use the parameters trained by Caffe [18] to initialize the AlexNet to accelerate the training. We use Gensim (`http://radimrehurek.com/gensim/`) to train the Skip-Gram model with the dimension of word vectors being 100

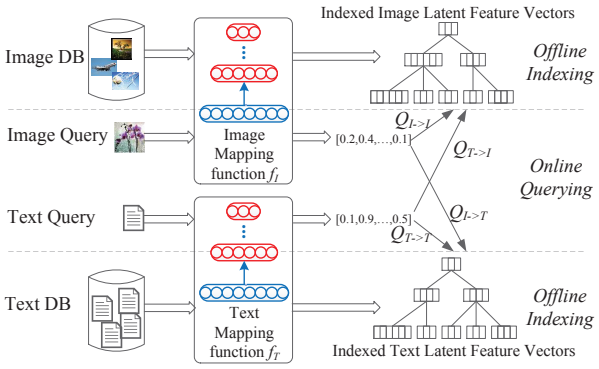


Fig. 8: Illustration of Query Processing

map it into the latent space through its modal-specific mapping function. Next, intra-modal and inter-modal searches are conducted against the corresponding index (i.e., the VA-File) shown in Figure 8. For example, the task of searching relevant tags of one image, i.e., $\mathbb{Q}_{I\rightarrow T}$, is processed by the index for the text latent vectors.

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. For example, in our experiment (Section 8.2), we use Spectral Hashing [41] , which converts real-valued vectors (data points) into binary codes with the objective to minimize the Hamming distance of data points that are close in the original Euclidean space. Other hashing approaches like [35, 12] are also applicable.

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.

7 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 [21, 36, 43]. Generally these papers 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 [21] extends the Spectral Hashing [41] 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 [36] learns the latent features of all training data firstly, which is computationally expensive. LCMH [43] 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 [4], MLBE [42] and LSCMR [25]. 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 correlations 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 [25] 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 (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 [29, 37] extends deep learning to multi-modal scenario. [37] combines two Deep Boltzmann Machines (DBM) (one for image, one for text) with a common latent layer to construct a Multi-modal DBM. [29] 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. DeViSE [9] from Google shares similar idea with our supervised training algorithm. They embed the image feature into text space. This embedded feature is then used to retrieve similar word vectors of labels for zero-shot learning. The embedding (mapping) function is learned against word vectors of labels. However for multi-modal retrieval, the text feature is generated from noisy tag vectors which are not considered when learning

the embedding function. Hence, it would be less effective than our supervised training algorithm which integrates label information into the learning objective. Our mapping functions would then generate more robust latent feature against noisy input data.

8 Experimental Study

This section provides an extensive performance study of our solution in comparison with the state-of-the-art methods. We examine both efficiency and effectiveness of our method including training overhead, query processing time and accuracy. Visualization of the training process is also provided to help understand the algorithms. All experiments are conducted on CentOS 6.4 using CUDA 5.5 with NVIDIA GPU (GeForce GTX TITAN). The size of main memory is 64GB and GPU memory is 6GB. The code and hyper-parameter settings are available online [7]. In the rest of this section, we give our evaluation metrics firstly, and then study the performance of unsupervised approach and supervised approach respectively.

8.1 Evaluation Metrics

We evaluate the effectiveness of the mapping mechanism by measuring the effectiveness of the multi-modal search, i.e., $\mathbb{Q}_{q\to t}(q, t \in \{T, I\})$, using the mapped latent features. Without specifications, searches are conducted against real-valued latent features using Euclidean distance. We use the Mean Average Precision (MAP) [27], one of the standard information retrieval metrics, as the major effectiveness evaluation metric. Given a set of queries, we calculate the Average Precision (AP) for each query q as,

$$AP(q) = \frac{\sum_{k=1}^{R} P(k)\delta(k)}{\sum_{j=1}^{R} \delta(j)} \tag{18}$$

where R is the size of the test dataset; $\delta(k) = 1$ if the k-th result is relevant, otherwise $\delta(k) = 0$; $P(k)$ 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 measure the *precision* and *recall* of search tasks. Given a query, the *ground truth* is defined as: *if a result shares at least one common label (or category) with the query, it is considered as a relevant result; otherwise it is irrelevant*.

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.

[7] http://www.comp.nus.edu.sg/~wangwei/code

Table 1: Statistics of Datasets for Unsupervised Training

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

8.2 Experimental Study of Unsupervised Approach

The datasets used for unsupervised training are described firstly. Then we analyze the training process by visualization. Comparison with previous works, including CVH [21], CMSSH [4] and LCMH [43] are provided afterwards. [8]

8.2.1 Datasets

Unsupervised training requires relevant image text pairs, which are easy to collect. We use three datasets to evaluate the performance—NUS-WIDE [5], Wiki [30] and Flickr1M [17].

NUS-WIDE The dataset contains 269,648 images from Flickr, each associated with 6 tags on average. We refer to the image and its tags as an image-text pair. There are 81 ground truth labels manually annotated for evaluation. Following previous works [24, 43], we extract 190,421 image-text pairs annotated with the most frequent 21 labels 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 are provided in the dataset [5]. 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 the 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 a label 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 represent one text section by 1,000

[8] The 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.

dimensional word count vector like [25]. The average number of words in one section is 131 which is much higher than that in NUS-WIDE. To avoid overflow in Equation 6 and smooth the text input, we normalize each unit x as $\log(x+1)$ [32].

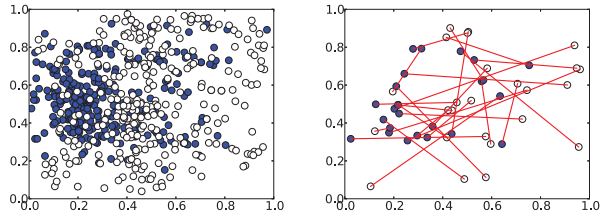
Flickr1M This dataset contains 1 million images associated with tags from Flickr. 25,000 of them are annotated with labels (there are 38 labels in total). The image feature is a 3,857 dimensional vector concatenated by SIFT feature, color histogram, etc [37]. Like NUS-WIDE, the text feature is represented 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 [19] to normalize each dimension of image feature to have zero mean and unit variance.

8.2.2 Training Visualization

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., β_I and β_T. 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 4 for details).

First, following the training procedure in Section 4, we train a 4-layer image SAE with the dimension of each layer as $500 \to 128 \to 16 \to 2$. Similarly, a 4-layer text SAE (the structure is $1000 \to 128 \to 16 \to 2$) is trained[9]. There is no standard guideline for setting the number of latent layers and units in each latent layer for deep learning [1]. In all our experiments, we adopt the widely used pyramid-like structure [15, 6], i.e. decreasing layer size from the bottom (or first hidden) layer to the top layer. In our experiment, we observed that 2 latent layers perform better than a single latent layer. But there is no significant improvement from 2 latent layers to 3 latent layers. Latent features of sampled image-text pairs from the validation set are plotted in Figure 9a. The 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. 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, as the SAEs are trained separately, 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 integrate the inter-modal loss in the learning objective as Equation 1. In the following figures, we only plot the distribution of these 25 pairs for ease of illustration.

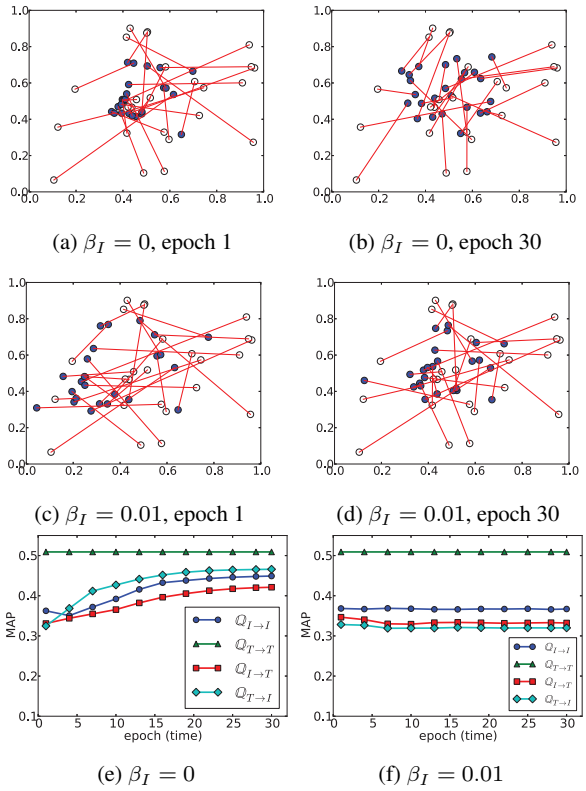


(a) 300 random image-text pairs (b) 25 image-text pairs

Fig. 9: Visualization of latent features after projecting them into 2D space (Blue points are image latent features; White points are text latent features. Relevant image-tex pairs are connected using red lines)

Second, we adjust the image SAE with the text SAE fixed 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 intuition in Section 3, we should use a small β_I to decrease the weight of image reconstruction error $\mathcal{L}_I$ in the objective function, i.e., Equation 1. To verify this, we compare the performance of two choices of β_I, namely $\beta_I = 0$ and $\beta_I = 0.01$. The first two rows of Figure 10 show the latent features generated by the image SAE after epoch 1 and epoch 30. Comparing image-text pairs in Figure 10b and 10d, we can see that with smaller β_I, the image latent features move closer to their relevant text latent features. This is in accordance with Equation 1, where smaller β_I relaxes the restriction on the image reconstruction error, and in turn increases the weight for inter-modal distance $\mathcal{L}_{I,T}$. By moving close to relevant text latent features, the image latent features gain more semantics. As shown in Figure 10e, the MAPs increase as training goes on. MAP of $\mathbb{Q}_{T\to T}$ does not change because the text SAE is fixed. When $\beta_I = 0.01$, the MAPs do not increase in Figure 10f. This is because image latent features hardly move close to the relevant text latent features as shown in Figure 10c and 10d.

[9] The last layer with two units is for visualization purpose, such that the latent features could be showed in a 2D space.

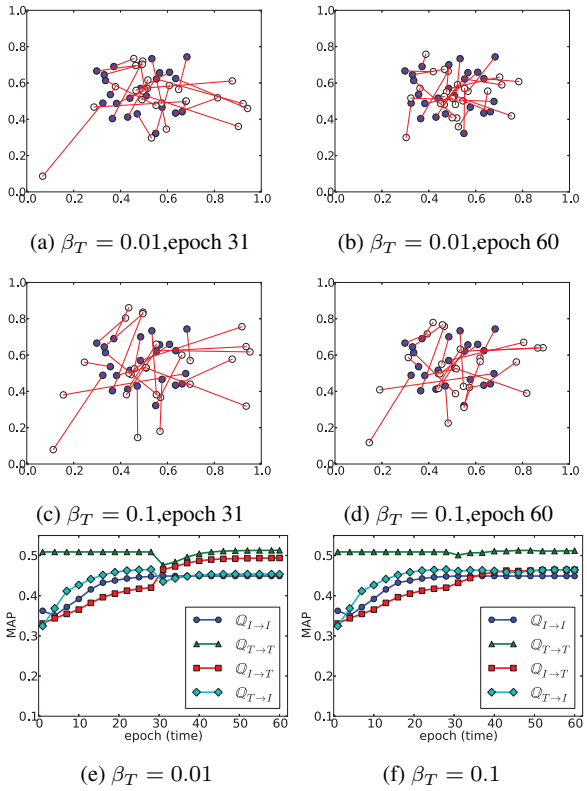


(a) $\beta_I = 0$, epoch 1 (b) $\beta_I = 0$, epoch 30

(c) $\beta_I = 0.01$, epoch 1 (d) $\beta_I = 0.01$, epoch 30

(e) $\beta_I = 0$ (f) $\beta_I = 0.01$

Fig. 10: Adjusting Image SAE with Different β_I and Text SAE fixed (a-d show the positions of features of image-text pairs in 2D space)



(a) $\beta_T = 0.01$,epoch 31 (b) $\beta_T = 0.01$,epoch 60

(c) $\beta_T = 0.1$,epoch 31 (d) $\beta_T = 0.1$,epoch 60

(e) $\beta_T = 0.01$ (f) $\beta_T = 0.1$

Fig. 11: Adjusting Text SAE with Different β_T and Image SAE fixed (a-d show the positions of features of image-text pairs in 2D space)

Third, we adjust the text SAE with the image SAE fixed from epoch 31 to epoch 60. We also compare two choices of β_T, namely 0.01 and 0.1. β_I is set to 0. 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 move much closer to the relevant image latent features. It leads to the big changes of MAPs at epoch 31 in Figure 11e. For example, $\mathbb{Q}_{T\to T}$ substantially drops from 0.5 to 0.46. This is because the sudden moves towards images change the intra-modal relationships of text latent features. Another big change happens on $\mathbb{Q}_{I\to T}$, whose MAP 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 move T towards I such that T is more likely to be the reverse nearest neighbor of I. Hence, the MAP of query $\mathbb{Q}_{I\to T}$ is greatly improved. On the contrary, $\mathbb{Q}_{T\to I}$ decreases. From epoch 32 to epoch 60, the text latent features on the one hand move close to relevant image latent features slowly, and on the other hand rebuild their intra-modal relationships. The latter is achieved by minimizing the reconstruction error $\mathcal{L}_T$ to capture the semantics of the original features. Therefore, both $\mathbb{Q}_{T\to T}$ and $\mathbb{Q}_{I\to T}$ grows gradually. Comparing Figure 11a and 11c, we can see the distance of relevant latent features in Figure 11c is larger than that in Figure 11a. The reason is that when β_T is larger, the objective function in Equation 1 pays more effort to minimize the reconstruction error $\mathcal{L}_T$. Consequently, less effort is paid to minimize the inter-modal distance $\mathcal{L}_{I,T}$. Hence, relevant inter-modal pairs cannot move closer. This effect is reflected as minor changes of MAPs at epoch 31 in Figure 11f in contrast with that in Figure 11e. Similarly, small changes happen between Figure 11c and 11d, which leads to minor MAP changes from epoch 32 to 60 in Figure 11f.

8.2.3 Evaluation of Model Effectiveness on NUS_WIDE Dataset

We first examine the mean average precision (MAP) of our method using Euclidean distance against 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 $\rightarrow$

Table 2: Mean Average Precision on NUS-WIDE dataset

Task		$\mathbb{Q}_{I\to I}$				$\mathbb{Q}_{T\to T}$				$\mathbb{Q}_{I\to T}$				$\mathbb{Q}_{T\to I}$			
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**

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

Task		$\mathbb{Q}_{I\to I}$				$\mathbb{Q}_{T\to T}$				$\mathbb{Q}_{I\to T}$				$\mathbb{Q}_{T\to I}$			
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	0.357	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**

$128 \to L$ by the text SAE. β_I and β_T are set to 0 and 0.01 respectively according to Section 8.2.2. 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 four search tasks. It demonstrates an average improvement of 17%, 27%, 21%, and 26% for $\mathbb{Q}_{I\to I}$, $\mathbb{Q}_{T\to T}$,$\mathbb{Q}_{I\to T}$, and $\mathbb{Q}_{T\to I}$ respectively. CVH and CMSSH prefer smaller L in queries $\mathbb{Q}_{I\to T}$ and $\mathbb{Q}_{T\to I}$. 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 thus is more robust under different L.

Figure 12 shows the precision-recall curves, and the recall-candidates ratio curves (used by [42, 43]) which show the change of recall when inspecting more results on the returned rank list. Due to space limitation, we only show the results of $\mathbb{Q}_{T\to I}$ and $\mathbb{Q}_{I\to T}$. We achieve similar trends on results of $\mathbb{Q}_{T\to T}$ and $\mathbb{Q}_{I\to I}$. Our method shows the best accuracy except when recall is 0 [10], 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 recall-candidates ratio, the curve of MSAE is always above those of other methods. It shows that we get better recall when inspecting the same number of objects. In other words, our method ranks more relevant objects at higher (front) positions. Therefore, MSAE performs better than other methods.

Besides real-valued features, we also conduct experiments against binary latent features for which Hamming distance is used as the distance function. In our implementation, we choose Spectral Hashing [41] 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 scores using Hamming distance is not as good as that of Euclidean distance. This is due to the possible information loss by converting real-valued features into binary features.

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

8.2.4 Evaluation of Model Effectiveness on Wiki Dataset

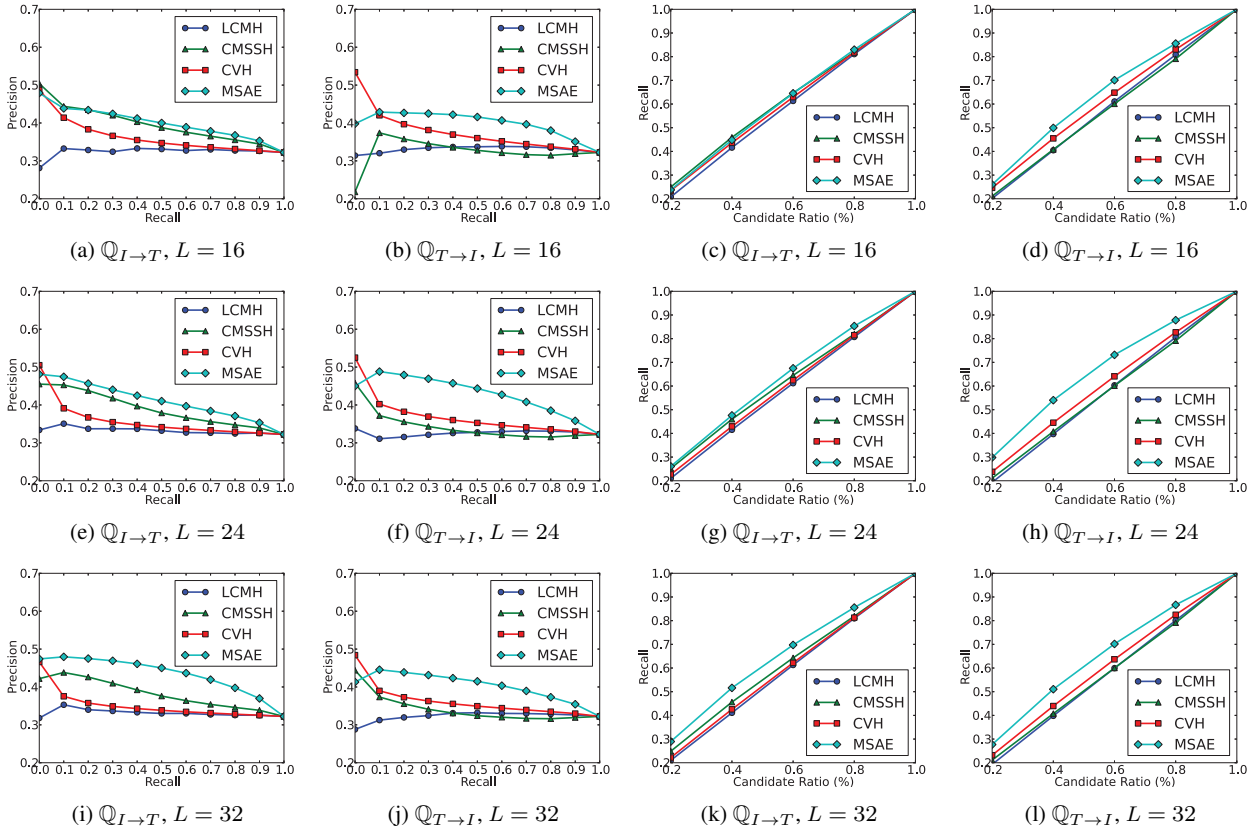
We conduct similar evaluations on **Wiki** dataset as on NUS-WIDE. For MSAE with latent feature of dimension L, the structure of its image SAE is $128 \to 128 \to L$, and the structure of its text SAE is $1000 \to 128 \to L$. Similar to the settings on NUS-WIDE, β_I is set to 0 due to the low quality of image features, and β_T is set to 0.01 to make $\mathcal{L}_T$ and $\mathcal{L}_{I,T}$ within the same scale.

The performance is reported in Table 4. MAPs on **Wiki** dataset are much smaller than those on NUS-WIDE except for $\mathbb{Q}_{T\to T}$. 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 within images, and between images and text. Query task $\mathbb{Q}_{T\to T}$ is not affected as Wkipedia'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 $\mathbb{Q}_{I\to I}$, $\mathbb{Q}_{T\to T}$,$\mathbb{Q}_{I\to T}$, and $\mathbb{Q}_{T\to I}$ 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.

8.2.5 Evaluation of Model Effectiveness on Flickr1M Dataset

We configure a 4-layer image SAE as $3857 \to 1000 \to 128 \to L$, and a 4-layer text SAE as $2000 \to 1000 \to 128 \to L$ for this dataset. Different from the other two datasets, the original image feature of Flickr1M are of higher quality as it consists of both local and global features. For intra-modal search, the image latent feature performs equally well as the text latent feature. Therefore, we set both β_I and β_T to 0.01.

We compare the MAP of MSAE and CVH in Table 5. MSAE outperforms CVH in most of the search tasks. The



(a) $\mathbb{Q}_{I\to T}$, $L=16$ (b) $\mathbb{Q}_{T\to I}$, $L=16$ (c) $\mathbb{Q}_{I\to T}$, $L=16$ (d) $\mathbb{Q}_{T\to I}$, $L=16$

(e) $\mathbb{Q}_{I\to T}$, $L=24$ (f) $\mathbb{Q}_{T\to I}$, $L=24$ (g) $\mathbb{Q}_{I\to T}$, $L=24$ (h) $\mathbb{Q}_{T\to I}$, $L=24$

(i) $\mathbb{Q}_{I\to T}$, $L=32$ (j) $\mathbb{Q}_{T\to I}$, $L=32$ (k) $\mathbb{Q}_{I\to T}$, $L=32$ (l) $\mathbb{Q}_{T\to I}$, $L=32$

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

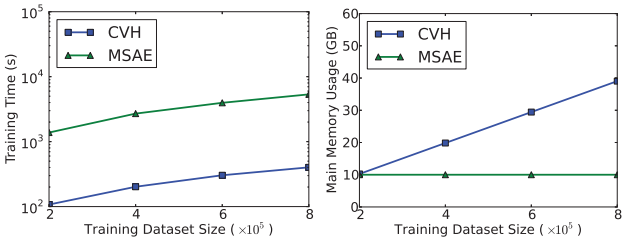
Table 4: Mean Average Precision on **Wiki** dataset

Task		$\mathbb{Q}_{I\to I}$				$\mathbb{Q}_{T\to T}$				$\mathbb{Q}_{I\to T}$				$\mathbb{Q}_{T\to I}$			
Algorithm		LCMH	CMSSH	CVH	MSAE	LCMH	CMSSH	CVH	MSAE	LCMH	CMSSH	CVH	MSAE	LCMH	CMSSH	CVH	MSAE
Dimension of Latent Space L	16	0.146	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**

results of LCMH and CMSSH cannot be reported as both methods run out of memory in the training stage.

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

Task		$\mathbb{Q}_{I\to I}$		$\mathbb{Q}_{T\to T}$		$\mathbb{Q}_{I\to T}$		$\mathbb{Q}_{T\to I}$	
Algorithm		CVH	MSAE	CVH	MSAE	CVH	MSAE	CVH	MSAE
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**



(a) (b)

Fig. 13: Training Cost Comparison on Flickr1M Dataset

8.2.6 Evaluation of Training Cost

We use the largest dataset Flickr1M to evaluate the training cost of time and memory consumption. The results are reported 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 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 about the number of training iterations times the height of MSAE. Possible solutions

for accelerating the MSAE training include adopting Distributed deep learning [7]. 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. 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. The actual minimum memory usage for MSAE is smaller than 10GB. In our experiments, we allocate more space to load multiple mini-batches into memory to save disk reading cost. CVH has to load all training data into memory for matrix operations. Therefore, its memory usage increases with respect to the size of the training dataset.

8.2.7 Evaluation of Query Processing Efficiency

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 dimension. 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 [40]) for each modality. For binary latent features, we do not create any indexes, as linear scan offers decent performance as shown in Figure 14. It shows the time of searching 50 nearest neighbors (averaged over 100 random queries) against datasets represented using binary latent features (based on Hamming distance) and real-valued features (based on Euclidean distance) respectively. We can see that 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.

8.3 Experimental Study of Supervised Approach

8.3.1 Datasets

Supervised training requires input image-text pairs to be associated with additional semantic labels. Since Flickr1M does not have labels and Wiki dataset has too few labels that are not discriminative enough, we use NUS-WIDE dataset to evaluate the performance of supervised training. We extract $203,400$ labelled pairs, among which $150,000$ are used for training. The remaining pairs are evenly partitioned into two sets for validation and testing. From both sets, we randomly select 2000 pairs as queries. This labelled dataset is named NUS-WIDE-a.

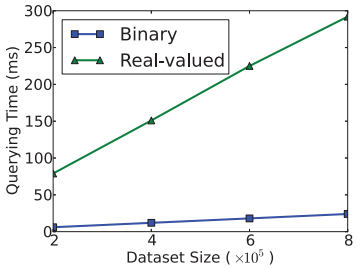


Fig. 14: Querying Time Comparison Using Real-valued and Binary Latent Features

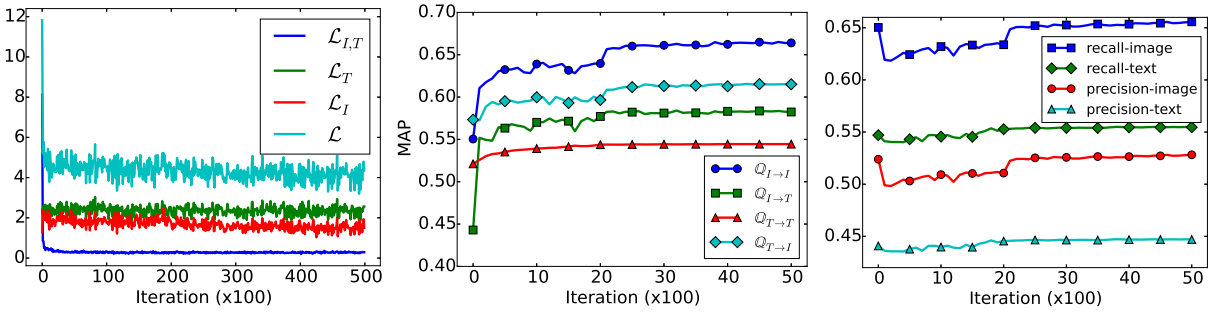
Table 6: Statistics of Datasets for Supervised Training

Dataset	**NUS-WIDE-a**	**NUS-WIDE-b**
Total size	203, 400	76,000
Training set	150,000	60,000
Validation set	26,700	80,000
Test set	26,700	80,000

We further extract another dataset from NUS-WIDE-a by filtering those pairs with more than one label. This dataset is named NUS-WIDE-b and is used to compare with DeViSE [9], which is designed for training against images annotated with single label. In total, we obtain $76,000$ pairs. Among them, we randomly select $60,000$ pairs for training and the rest are evenly partitioned for validation and testing. 1000 queries are randomly selected from the two datasets respectively.

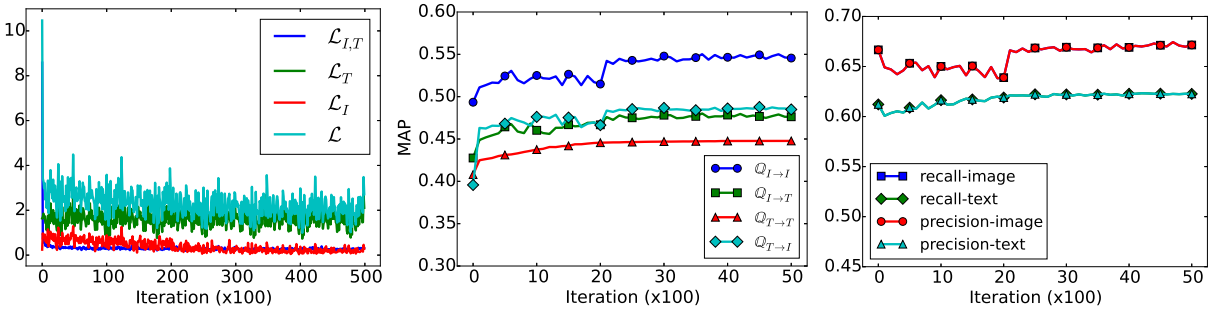
8.3.2 Visualization of Training Process

NUS-WIDE-a In Figure 15a, we plot the total training loss $\mathcal{L}$ and its components ($\mathcal{L}_I$, $\mathcal{L}_T$ and $\mathcal{L}_{I,T}$) in the first $50,000$ iterations (one iteration for one mini-batch) against the NUS-WIDE-a dataset. We can see that training converges rather quickly. The training loss drops dramatically at the very beginning and then decreases slowly. This is because initially the learning rate is large and the parameters approach quickly towards the optimal values. Another observation is that the intra-modal loss $\mathcal{L}_I$ for the image modality is smaller than $\mathcal{L}_T$ for the text modality. This is because some tags may be noisy or not very relevant to the associated labels that represent the main visual content in the images. It is difficult to learn a set of parameters to map noisy tags into the latent space and well predict the ground truth labels. The inter-model training loss is calculated at a different scale and is



(a) Training Loss (b) MAP on Validation Dataset (c) Precision-Recall on Validation Dataset

Fig. 15: Visualization of Training on NUS-WIDE-a



(a) Training Loss (b) MAP on Validation Dataset (c) Precision-Recall on Validation Dataset

Fig. 16: Visualization of Training on NUS-WIDE-b

normalized to be within one order of magnitude as $\mathcal{L}_I$ and $\mathcal{L}_T$.

The MAPs for all types of searches using supervised training model are shown in Figure 15b. As can be seen, the MAPs first gradually increase and then become stable in the last few iterations. It is worth noting that the MAPs are much higher than the results of unsupervised training (MSAE) in Figure 11. There are two reasons for the superiority. First, the supervised training algorithm (MDNN) exploits DCNN and NLM to learn better visual and text features respectively. Second, labels bring in more semantics and make latent features learned more robust to noises in input data (e.g., visual irrelevant tags).

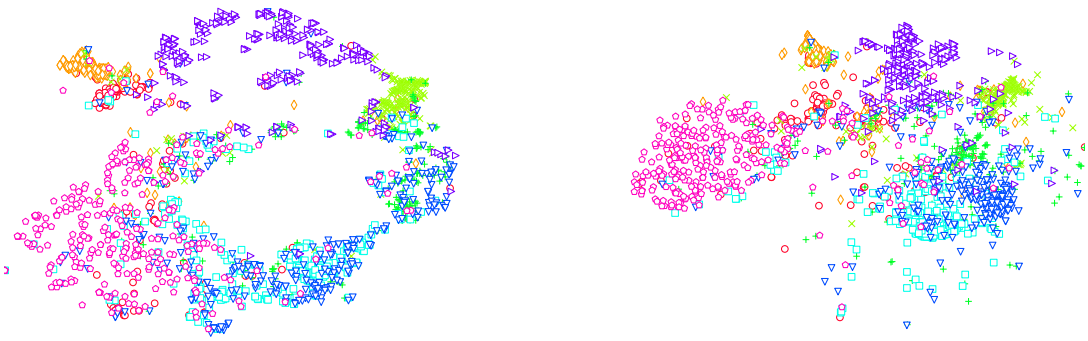
Besides MAP, we also evaluate MDNN by comparing it with [11] for multi-label prediction based on precision and recall. For each image (or text), we look at its labels with the largest k probabilities based on Equation 9 (or 16). For the i-th image (or text), let N_i denote the number of labels out of k that belong to its ground truth label set, and T_i the size of its ground truth label set. The precision and recall are defined according to [11] as follows:

$$precison = \frac{\sum_{i=1}^{n} N_i}{k * n}, \quad recall = \frac{\sum_{i=1}^{n} N_i}{\sum_{i=1}^{n} T_i} \tag{19}$$

where n is the test set size. The results are shown in Figure 15c ($k = 3$). The performance decreases at the early stage and then goes up. This is because at the early stage, in order to minimize the inter-modal loss, the training may disturb the pre-trained parameters fiercely that affects the intra-modal search performance. Once the inter-modal loss is reduced to a certain level, it starts to adjust the parameters to minimize both inter-modal loss and intra-modal loss. Hence, the classification performance starts to increase. We can also see that the performance of latent text features is not as good as that of latent image features due to the noises in tags. We use the same experiment setting as that in [11], the (over all) precision and recall is 7% and 7.5% higher than that in [11] respectively.

NUS-WIDE-b Figure 16 shows the training results against the NUS-WIDE-b dataset. The results demonstrate similar patterns to those in Figure 15. However, MAPs become lower, possibly due to smaller training dataset size and fewer number of associated labels. In Figure 16c, the precision and recall for classification using image (or text) latent features are the same. This is because each image-text pair has only one label and $T_i = 1$. When we set $k = 1$, the denominator $k * n$ in precision is equal to $\sum_{i=1}^{n} T_i$ in recall.

2D Visualization To demonstrate that the learned mapping functions can generate semantic discriminative latent features, we extract top-8 most popular labels and for each label, we randomly sample 300 image-text pairs from the



(a) Image Latent Feature (b) Text Latent Feature

Fig. 17: Visualization of Latent Features Learned by MDNN for the Test Dataset of NUSWDIE-a (features represented by the same shapes and colors are annotated with the same label)

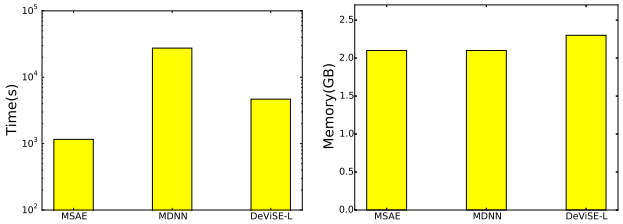
Table 7: Mean Average Precision using Real-valued Latent Feature

Task		$\mathbb{Q}_{I\to I}$			$\mathbb{Q}_{T\to T}$			$\mathbb{Q}_{I\to T}$			$\mathbb{Q}_{T\to I}$		
Algorithm		MDNN	DeViSE-L	DeViSE-T	MDNN	DeViSE-L	DeViSE-T	MDNN	DeViSE-L	DeViSE-T	MDNN	DeViSE-L	DeViSE-T
Dataset	NUS-WIDE-a	**0.669**	0.5619	0.5399	**0.541**	0.468	0.464	**0.587**	0.483	0.517	**0.612**	0.502	0.515
	NUS-WIDE-b	**0.556**	0.432	0.419	**0.466**	0.367	0.385	**0.497**	0.270	0.399	**0.495**	0.222	0.406

test dataset of NUS-WIDE-b. Their latent features are projected into a 2-dimensional space by t-SNE [26]. Figure 17a shows the 2-dimensional image latent features where one point represents one image feature and Figure 17b shows the 2-dimensional text features. Labels are distinguished using different shapes. We can see that the features are well clustered according to their labels. Further, the image features and text features semantically relevant to the same labels are projected to similar positions in the 2D space. For example, in both figures, the red circles are at the left side, and the blue right triangles are in the top area. The two figures together confirm that our supervised training is very effective in capturing semantic information for multi-modal data.

8.3.3 Evaluation of Model Effectiveness on NUS-WIDE Dataset

In our final experiment, we compare MDNN with DeViSE [9] in terms of effectiveness of multi-modal retrieval. DeViSE maps image features into text feature space. The learning objective is to minimize the rank hinge loss based on the latent features of an image and its labels. We implement this algorithm and extend it to handle multiple labels by averaging their word vector features. We denote this algorithm as DeViSE-L. Besides, we also implement a variant of DeViSE denoted as DeViSE-T, whose learning objective is to minimize the rank hinge loss based on the latent features of an image and its tag(s). Similarly, if there are multiple tags,



(a) Training Time (b) Memory Consumption

Fig. 18: Training Cost Comparison on NUSWIDE-a Dataset

we average their word vectors. The results are shown in Table 7. The retrieval is conducted using real-valued latent feature and cosine similarity as the distance function. We can see that MDNN performs much better than both DeViSE-L and DeViSE-T for all four types of searches on both NUS-WIDE-a and NUS-WIDE-b. The main reason is that the image tags are not all visually relevant, which makes it hard for the text (tag) feature to capture the visual semantics in DeViSE. MDNN exploits the label information in the training, which helps to train a model that can generate more robust feature against noisy input tags. Hence the performance of MDNN is better.

8.3.4 Evaluation of Training Cost

We report the training cost in terms of training time (Fig. 18a) and memory consumption (Fig. 18b) on NUS-WIDE-a dataset.

Training time includes the pre-training for each single modality and the joint multi-modal training. MDNN and DeViSE-L take longer time to train than MSAE, because the convolution operations in them are time consuming. Further, MDNN involves pre-training stages for the image modality and text modality, and thus incurs longer training time than DeViSE-L. The memory footprint of MDNN is similar to that of DeViSE-L, as the two methods both rely on DCNN, which incurs most of the memory consumption. DeViSE-L uses features of higher dimension (100 dimension) than MDNN (81 dimension), which leads to about 100 MegaBytes difference as shown in Fig. 18b.

8.3.5 Comparison with Unsupervised Approach

By comparing Table 7 and Table 2, we can see that the supervised approach–MDNN performs better than the unsupervised approach–MSAE. This is not surprising because MDNN consumes more information than MSAE. Although the two methods share the same general training objective, the exploitation of label semantics helps MDNN learn better features in capturing the semantic relevance of the data from different modalities. For memory consumption, MDNN and MSAE perform similarly (Fig. 18b).

9 Conclusion

In this paper, we have proposed a general framework (objective) for learning mapping functions for effective multi-modal retrieval. Both intra-modal and inter-modal semantic relationships of data from heterogeneous sources are captured in the general learning objective function. Given this general objective, we have implemented one unsupervised training algorithm and one supervised training algorithm separately to learn the mapping functions based on deep learning techniques. The unsupervised algorithm uses stacked auto-encoders as the mapping functions for the image modality and the text modality. It only requires simple image-text pairs for training. The supervised algorithm uses an extend DCNN as the mapping function for images and an extend NLM as the mapping function for text data. Label information is integrated in the training to learn robust mapping functions against noisy input data. The results of experiment confirm the improvements of our method over previous works in search accuracy. Based on the processing strategies outlined in this paper, we have built a distributed training platform (called SINGA) to enable efficient deep learning training that supports training large scale deep learning models. We shall report the system architecture and its performance in a future work.

10 Acknowledgments

This work is supported by A*STAR project 1321202073. Xiaoyan Yang is supported by Human-Centered Cyber-physical Systems (HCCS) programme by A*STAR in Singapore.

References

1. Bengio Y (2012) Practical recommendations for gradient-based training of deep architectures. CoRR abs/1206.5533
2. Bengio Y, Ducharme R, Vincent P, Janvin C (2003) A neural probabilistic language model. Journal of Machine Learning Research 3:1137–1155
3. Bengio Y, Courville AC, Vincent P (2013) Representation learning: A review and new perspectives. IEEE Trans Pattern Anal Mach Intell 35(8):1798–1828
4. Bronstein MM, Bronstein AM, Michel F, Paragios N (2010) Data fusion through cross-modality metric learning using similarity-sensitive hashing. In: CVPR, pp 3594–3601
5. Chua TS, Tang J, Hong R, Li H, Luo Z, Zheng YT (July 8-10, 2009) 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.
6. Ciresan DC, Meier U, Gambardella LM, Schmidhuber J (2012) Deep big multilayer perceptrons for digit recognition. vol 7700, Springer, pp 581–598
7. Dean J, Corrado G, Monga R, Chen K, Devin M, Le QV, Mao MZ, Ranzato M, Senior AW, Tucker PA, Yang K, Ng AY (2012) Large scale distributed deep networks. In: NIPS Lake Tahoe, Nevada, United States., pp 1232–1240
8. Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, Darrell T (2013) Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:13101531
9. Frome A, Corrado GS, Shlens J, Bengio S, Dean J, Ranzato M, Mikolov T (2013) Devise: A deep visual-semantic embedding model. In: NIPS, Lake Tahoe, Nevada, United States., pp 2121–2129
10. Girshick RB, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR 2014, Columbus, OH, USA, June 23-28, 2014, pp 580–587
11. Gong Y, Jia Y, Leung T, Toshev A, Ioffe S (2013) Deep convolutional ranking for multilabel image annotation. CoRR abs/1312.4894
12. Gong Y, Lazebnik S, Gordo A, Perronnin F (2013) Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans Pattern Anal Mach Intell 35(12):2916–2929

13. Goroshin R, LeCun Y (2013) Saturating auto-encoder. CoRR abs/1301.3577
14. Hinton G (2010) A Practical Guide to Training Restricted Boltzmann Machines. Tech. rep.
15. Hinton G, Salakhutdinov R (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504 – 507
16. Hjaltason GR, Samet H (2003) Index-driven similarity search in metric spaces. ACM Trans Database Syst 28(4):517–580
17. Huiskes MJ, Lew MS (2008) The mir flickr retrieval evaluation. In: Multimedia Information Retrieval, pp 39–43
18. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick RB, Guadarrama S, Darrell T (2014) Caffe: Convolutional architecture for fast feature embedding. In: MM'14, Orlando, FL, USA, 2014, pp 675–678
19. Krizhevsky A (2009) Learning multiple layers of features from tiny images. Tech. rep.
20. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25, pp 1106–1114
21. Kumar S, Udupa R (2011) Learning hash functions for cross-view similarity search. In: IJCAI, pp 1360–1365
22. LeCun Y, Bottou L, Orr G, Müller K (1998) Efficient BackProp. In: Orr G, Müller KR (eds) Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, vol 1524, Springer Berlin Heidelberg, Berlin, Heidelberg, chap 2, pp 9–50
23. Liu D, Hua X, Yang L, Wang M, Zhang H (2009) Tag ranking. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009, pp 351–360, DOI 10.1145/1526709.1526757
24. Liu W, Wang J, Kumar S, Chang SF (2011) Hashing with graphs. In: ICML, pp 1–8
25. Lu X, Wu F, Tang S, Zhang Z, He X, Zhuang Y (2013) A low rank structural large margin method for cross-modal ranking. In: SIGIR, pp 433–442
26. van der Maaten L (2014) Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15:3221–3245
27. Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press
28. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. CoRR abs/1301.3781
29. Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY (2011) Multimodal deep learning. In: ICML, pp 689–696
30. Rasiwasia N, Pereira JC, Coviello E, Doyle G, Lanckriet GRG, Levy R, Vasconcelos N (2010) A new approach to cross-modal multimedia retrieval. In: ACM Multimedia, pp 251–260
31. Rifai S, Vincent P, Muller X, Glorot X, Bengio Y (2011) Contractive auto-encoders: Explicit invariance during feature extraction. In: ICML, pp 833–840
32. Salakhutdinov R, Hinton GE (2009) Semantic hashing. Int J Approx Reasoning 50(7):969–978
33. Socher R, Manning CD (2013) Deep learning for NLP (without magic). In: Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA, pp 1–3
34. Socher R, Pennington J, Huang EH, Ng AY, Manning CD (2011) Semi-supervised recursive autoencoders for predicting sentiment distributions. In: EMNLP, pp 151–161
35. Song J, Yang Y, Huang Z, Shen HT, Hong R (2011) Multiple feature hashing for real-time large scale near-duplicate video retrieval. In: MM, 2011, ACM, pp 423–432
36. Song J, Yang Y, Yang Y, Huang Z, Shen HT (2013) Inter-media hashing for large-scale retrieval from heterogeneous data sources. In: SIGMOD Conference, pp 785–796
37. Srivastava N, Salakhutdinov R (2012) Multimodal learning with deep boltzmann machines. In: NIPS, pp 2231–2239
38. Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: ICML, pp 1096–1103
39. Wang W, Ooi BC, Yang X, Zhang D, Zhuang Y (2014) Effective multi-modal retrieval based on stacked auto-encoders. PVLDB 7(8):649–660
40. Weber R, Schek HJ, Blott S (1998) A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB, pp 194–205
41. Weiss Y, Torralba A, Fergus R (2008) Spectral hashing. In: NIPS, pp 1753–1760
42. Zhen Y, Yeung DY (2012) A probabilistic model for multimodal hash function learning. In: KDD, pp 940–948
43. Zhu X, Huang Z, Shen HT, Zhao X (2013) Linear cross-modal hashing for efficient multimedia search. In: ACM Multimedia Conference, MM '13, Barcelona, Spain, October 21-25, 2013, pp 143–152
44. Zhuang Y, Yang Y, Wu F (2008) Mining semantic correlation of heterogeneous multimedia data for cross-media retrieval. IEEE Transactions on Multimedia 10(2):221–229

A Appendix

Algorithm 1 MiniBatchSGD(D, b, f, α)

Input: D, training dataset
Input: b, batchsize
Input: f, the initial mapping function
Input: α, learning rate//set manually
Output: f, updated mapping function.

1. $\theta \leftarrow f$//initial parameters of f
2. **repeat**
3. **for** $i = 0$ to $|D|/b$ **do**
4. $B = D[i * b : i * b + b]$
5. $\nabla\theta$=average(**BackPropogation**(x, f) for $x \in B$)
6. $\theta = \theta + \alpha * \nabla\theta$
7. **until** Converge
8. return f

Algorithm 2 BackPropagation(x_0, f)

Input: x_0, input feature vector
Input: f, the mapping function with parameter set θ
Output: θ, gradients of parameters of f

1. $h \leftarrow$ height of layers in f
2. $\{x_i\}_{i=1}^{h} \leftarrow f(x_0|\theta)$// forward through all layers
3. $\delta_h = \frac{\partial \mathcal{L}}{\partial x_h}$
4. **for** $i = h$ to 1 **do**
5. $\nabla\theta_i = \delta_i * \frac{\partial x_i}{\partial \theta_i}$// $\frac{\partial \mathcal{L}}{\partial \theta_i}$
6. $\delta_{i-1} = \delta_i * \frac{\partial x_i}{\partial x_{i-1}}$
7. return $\{\nabla\theta_i\}_{i=1}^{h}$

In this section, we present the mini-batch Stochastic Gradient Descent (mini-batch SGD) algorithm and the Back-Propagation (BP) algorithm [22], which are used throughout this paper to train MSAE and MDNN.

Mini-batch SGD minimizes the objective loss (e.g., $\mathcal{L}, \mathcal{L}_I, \mathcal{L}_T$) by updating the parameters involved in the mapping function(s) based on the gradients of the objective w.r.t the parameters. Specifically, it iterates the whole dataset to extract mini-batches (Line 4). For each mini-batch, it averages the gradients computed from BP (Line 5), and updates the parameters (Line 6).

BP calculates the gradients of the objective loss (e.g., $\mathcal{L}$, $\mathcal{L}_I$, $\mathcal{L}_T$) w.r.t. the parameters involved in the mapping function (e.g., f_I, f_T) using a chain rule (Equation 20, 21). It forwards the input feature vector through all layers of the mapping function (Line 2). Then it backwards the gradients according the chain rule (Line 4-6). θ_i denotes parameters involved in the i-th layer. Gradients are returned at Line 7.

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{\partial \mathcal{L}}{\partial x_i} * \frac{\partial x_i}{\partial \theta_i} \tag{20}$$

$$\frac{\partial \mathcal{L}}{\partial x_{i-1}} = \frac{\partial \mathcal{L}}{\partial x_i} * \frac{\partial x_i}{\partial x_{i-1}} \tag{21}$$