GrowBit: Incremental Hashing for Cross-Modal Retrieval

Devraj Mandal[1], Yashas Annadani[2], and Soma Biswas[1]

[1] Indian Institute of Science, Bangalore
[2] ETH Zurich
devrajm@iisc.ac.in, ayashas@student.ethz.ch,
somabiswas@iisc.ac.in

Abstract. Cross-modal retrieval using hashing techniques is gaining increasing importance due to its efficient storage, scalability and fast query processing speeds. In this work, we address a related and relatively unexplored problem: given a set of cross-modal data with their already learned hash codes, can we increase the number of bits to better represent the data without relearning everything? This scenario is especially important when the number of tags describing the data increases, necessitating longer hash codes for better representation. To tackle this problem, we propose a novel approach called *GrowBit*, which incrementally learns the bits in the hash code and thus utilizes all the bits learned so far. We develop a two-stage approach for learning the hash codes and hash functions separately, utilizing a recent formulation which decouples over the bits so that it can incorporate the incremental approach. Experiments on MirFlickr, IAPR-TC-12 and NUS-WIDE datasets show the usefulness of the proposed approach.

Keywords: Cross-modal retrieval · Hashing · Incremental Learning.

1 Introduction

Due to the availability of large amounts of multimedia data, cross-modal retrieval has become an active area of research [5], [8], [15]. For example, given an image query, it is often required to retrieve relevant textual documents from the database. Hashing techniques designed to generate good binary encodings for capturing the semantic relations between the data have gained popularity because of their impressive retrieval results, low-storage costs and efficient retrieval. Hashing techniques for both unsupervised [36], [23], [5], [35] and supervised settings [1], [13], [36], [31] have been proposed.

Multimedia data is often described using multiple tags (or labels or attributes) which gives a richer description of the data. Some illustrative examples of cross-modal multimedia data are shown in Figure 1. Since for training, the data is usually manually annotated or requires manual supervision, getting all the tags for the data at once might not be feasible due to limited human resources. Let us consider that the training data $\{X_{tr}, Y_{tr}\}$ has been annotated with L_{tr} tags, using which we learn a hash bit representation of k_1 bits. Gradually, with more human resources, more annotations of the data become available. Let us assume that now a total of $\hat{L}_{tr}$ tags become available. Consider an example, where, an existing dress catalogue which has been stored based on

Fig. 1. Some illustrative image-text data pairs from the three datasets - MirFlickr [10] (left), NUS-WIDE [3] (middle) and IAPR TC-12 [6] (right).

their category (like shirt, top) may now be annotated with finer details like sleeve length, etc., which requires a better representation using more bits so that it can be correctly retrieved using a textual description. Most of the current algorithms require relearning the entire set of hash codes and hash functions. In this work, we propose a novel incremental hashing approach termed *GrowBit*, which can efficiently utilize the already learned hash codes, and incrementally learn the additional bits, without compromising on the retrieval performance.

Inspired by the success of the two-stage hashing approaches [20], [15], [14], [27], we design our *GrowBit* as a two-stage approach, in which the additional hash bits are learned in the first stage and the new hash functions are learned in the second stage based on these additional hash bits. More specifically, we build upon the formulation of a recent state-of-the-art technique [20] which decouples over the bit representation and allows us to learn the bits in an incremental manner. In the first stage, in order to utilize the already learned hash codes, we compute the additional bits such that they learn the semantic information not captured by the initially learned bits. We utilize a deep neural network to learn the respective hash functions in the second stage. To learn the hash functions for the newly learned additional bits, we efficiently re-utilize parts of the network which was previously trained and modify it accordingly. This has an added advantage of significantly reducing the training time as compared to completely retraining the model. In addition, we propose a novel unification scheme to generate common hash codes using complementary information from multiple modalities. We perform an exhaustive evaluation on three standard cross-modal datasets, namely MirFlickr [10], IAPR-TC-12 [6] and NUS-WIDE [3] and show that the proposed algorithm can efficiently learn the hash codes under different scenarios. The contributions of this work are as follows:

1. We propose a novel incremental hashing approach *GrowBit*, which can seamlessly integrate the increasing tags in an incremental fashion, while utilizing the already learned hash code representations. To the best of our knowledge, this is the first work on incremental hashing in a cross-modal setting.
2. We also propose a unification strategy to generate common hash codes for representation.

3. The proposed approach results in a significant decrease in the number of new parameters to be trained in the second stage without compromising much on the retrieval performance.

2 Related Work

For cross-modal retrieval, algorithms to learn binary embeddings have been developed using both unsupervised and supervised techniques. Unsupervised approaches like [36], [23], [5], [35] uses the criterion of inter and intra affinity preservation for design of the binary latent embedding space. The work in CMFH [5] uses matrix factorization to learn a unified hash code. A recent method, inspired from dictionary learning, called quantization, has been proposed in [26], [18], [34] and shows good promise in retrieval performance.

Supervised approaches use label information to learn effective and more discriminative binary embedding like CMSSH [1] and CVH [13]. Incorporating label information in a matrix factorization based approach is proposed in [19]. In addition, the work in [19] is capable of handling large amounts of data seamlessly. SePH [15] converts the label information into probability distribution and uses the criterion of low Kullback-Leibler divergence to learn the bits. It also uses a unification stage which further boosts the retrieval performance. A generalized method to handle both paired and unpaired data in cross-modal settings with unification has been proposed in [20]. In [17], in the learning phase, each bit of the code can be sequentially learned with a discrete optimization scheme that jointly minimizes the empirical loss based on a boosting strategy. In a bitwise manner, hash functions are then learned for each modality, mapping the corresponding representations into unified hash codes [17]. The discriminative capability of the labels has been used to learn the hash code and hash function in [29]. Furthermore, care has been taken to suitably design constraints for the objective function to reduce the quantization error. Deep learning techniques for cross-modal hashing has been developed in [11]. The deep learning models [11] have shown major improvements whenever end-to-end training network with non-linear hash functions have been used. The semantic structure of the multi-modal data has been effectively captured by a deep generative framework in [32].

The problem of online hashing [28], [7] studies models which can deal with new incoming data while retaining important information learned from all previous data. This work considers the problem of adapting an already learned model to incorporate the ever-expanding set of tags (or attributes or descriptions) of data samples whereas, in an online setting, the incoming data is usually spread across the same set of categories as the previously available examples. Sequential learning of hash codes have been proposed in the works of [31], [25] where hash functions are learned incrementally to correct the errors made by the previous bits sequentially. However, those methods [25], [31] uses linear transformations to learn hash functions which usually gives poor retrieval performance as compared to non-linear based methods.

In this work, for developing an incremental hashing approach, we draw inspiration from the recent two-stage approaches [14], [27], [15]. Instead of learning the optimal hash codes and mapping functions in a joint optimization framework, the two-stage

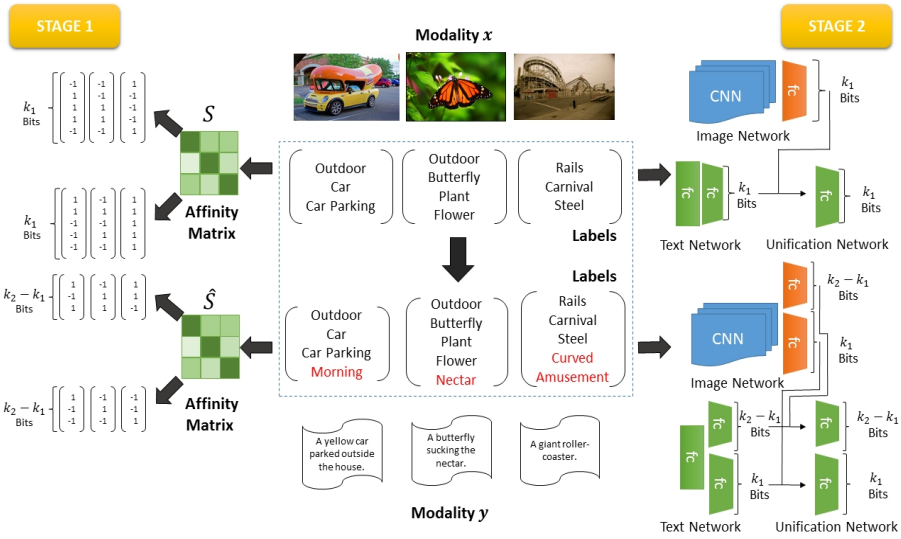

Fig. 2. An illustration of the proposed approach.

approaches first learn the hash codes followed by learning the hash functions, which often lead to simpler formulations. A schematic of the proposed algorithm is shown in Figure 2. Here, the affinity information (S) encoded in the labels is decomposed to generate the hash codes and a deep neural network (DNN) is used to learn the hash functions. A change in the affinities ($\hat{S}$, provided by the new tags highlighted in red) can be captured by learning the extended hash codes. The corresponding hash functions can then be learned by reusing parts of the previous DNN resulting in significant savings in computation and time without adversely affecting the retrieval performance. The work in [2] came to our notice recently and we believe that both the works help to advance the state-of-the-art in hashing. The main differences with GrowBit (vs [2]) are - (1) usage of standard hamming distance (weighted hamming metric) (2) ability to learn extendable hash bits (3) a novel unification scheme for cross-modal retrieval and (4) use of mean square loss (hinge loss) to learn the hash functions.

3 Proposed Approach

Problem Definition and Notation: Let the training data for the two modalities be denoted as $X_{tr} \in \mathbb{R}^{N \times d_x}$ and $Y_{tr} \in \mathbb{R}^{N \times d_y}$, where N is the number of training examples and d_x, d_y are the dimensionality of the data (in general $d_x \neq d_y$). Suppose, the initial set of tags available is denoted as $L_{tr} \in [0,1]^{N \times a}$, where a is the total number of tags and $(0/1)$ denotes the (absence/presence) of the individual tag. The already learned hash code representations for X_{tr} and Y_{tr} are denoted as $H^x \in [-1,1]^{N \times k_1}$ and $H^y \in [-1,1]^{N \times k_1}$, where k_1 is the number of bits in the hash codes. Let $\hat{L}_{tr} \in [0,1]^{N \times \hat{a}}$ denote the new set of tags, where $\hat{a}$ denotes the increased set of tags. As a result of

more number of tags, the data has richer representation. We would like to represent this data with a richer representation using hash codes with more number of bits (say k_2), which should result in better cross-modal retrieval. Even if the number of tags remain the same, we may want to increase the number of bits to better represent the data for improved retrieval performance. One option is to learn all the k_2 bits from scratch. In this work, we explore whether we can utilize the already learned k_1 bits, and only learn the additional $(k_2 - k_1)$ bits, such that they capture the semantic relation between the data not captured by the initial bits. Since we are not relearning everything, there is significant savings in computation. In the second stage, we utilize deep learning architectures in a block-wise fashion to learn the hash functions for the additional hash bits.

In this work, X represents a matrix, x_{i*}, x_{*j} represents its i^{th} row, j^{th} column and x_{ij} represents its $(i,j)^{th}$ element. $\|\cdot\|_F$ denotes the Frobenius norm: $\|X\|_F^2 = \text{Tr}(X^T X)$. The operation $x_{ij}^* = \text{Proj}_{[-1,1]} x_{ij}$ can be defined as $x_{ij}^* = \{-1, x_{ij}, 1\}$; if, $\{x_{ij} < -1, x_{ij} \in [-1,1], x_{ij} > 1\}$ respectively. The sign operation is defined as $\text{sign}(x_{ij}) = 1$ if $x_{ij} \geq 0$, and $= -1$ otherwise. Proj and sign applied to vector and matrix respectively are done point-wise.

Motivation and Background: One way to design an incremental hash code learning framework is to utilize a formulation which decouples over the variables (bits) to be learned. In literature, such formulation can be found in [20], [33], [4], [38], which relate the similarity measure such that it can easily decouple over the bits. Here, the similarity between the i^{th} and j^{th} samples of the two modalities are measured using inner product of the k_1 bits i.e., $s_{ij} = h_{i*}^{x\,T} h_{j*}^{y}$, which is equivalent to $s_{ij} = \sum_{k=1}^{k_1} h_{ik}^x h_{jk}^y$, i.e. the objective remains decoupled over the bit dimension. Similarly, the new similarity $\hat{s}_{ij}$ can be expressed in terms of the expanded set of k_2 bits and can also be written in terms of the previously computed bits as

$$\hat{s}_{ij} = \sum_{k=1}^{k_1} h_{ik}^x h_{jk}^y + \sum_{k=k_1+1}^{k_2} \overline{h_{ik}^x}\,\overline{h_{jk}^y} \tag{1}$$

where the new hash bits $\overline{h_{ik}^x}, \overline{h_{jk}^y}$ of $(k_2 - k_1)$ length captures the difference in similarity between $\hat{s}_{ij}$ and s_{ij} that has not already been captured by the k_1 length hash bits h_{ik}^x, h_{jk}^y. [20] utilizes a similar formulation to learn the hash codes, but does not use any incremental updates.

Another such formulation can be found in [33], [4], [38] which deals with the triplet ranking loss whose general form is

$$\mathcal{L}\left(h_{i*}^x, h_{j*}^y, h_{k*}^y\right) = \max\left(0, 1 - \left(\|h_{i*}^x - h_{k*}^y\|_2^2 - \|h_{i*}^x - h_{j*}^y\|_2^2\right)\right) \tag{2}$$

where, $\{h_{i*}^x, h_{j*}^y, h_{k*}^y\}$ are the bit representations of three data samples. The squared norm is the sum of the square of the elements [9] ($||x||_2^2 = \sum_i x_i^2$) and hence the above objective can be expressed as a summation over the individual bits. The work in [38] utilizes this fact to design fast and efficient hash learning protocols for the single modal applications.

In this work, we build upon the first kind of formulation and show how it can be modified for incremental hashing. The second formulation is also suitable for a similar

approach. The formulations in [31], [25] can also be used to learn the hash functions sequentially. However, as the two stages are coupled together, the form of the hash function is limited to a linear transformation. Our two stage method removes this restriction and enables us to efficiently use DNN to learn better functions for hash bit representations. Also using DNN to learn Stage 2 allows us to re-use parts of the network to learn hash functions incrementally with considerable savings in time and computation.

3.1 Stage 1: Learning the Hash Code

We will first briefly describe the formulation in [20] and then extend it for our problem. During training, the binary representations of the i^{th} data of x modality and j^{th} data of y modality are computed in such a way that its similarity ($h_{i*}^{x\,T} h_{j*}^{y}$) is consistent with the semantic affinity given by their associated labels ($s_{ij} =< L_{tr}^{i}, L_{tr}^{j} >$). $< .,. >$ denotes the normalized inner product operation [15][20]. The two hash codes H^x and H^y (of k_1 dimension) are computed using the affinity matrix S by solving the following least square formulation [20], [27] which encodes all possible relations between the data using the initial labels L_{tr}

$$\begin{aligned} \min_{H^x, H^y} \quad & \|k_1 S - H^x H^{y\,T}\|_F^2 \\ \text{s.t.} \quad & H^x \in \{-1, 1\}^{N \times k_1}, \;\; H^y \in \{-1, 1\}^{N \times k_1}. \end{aligned} \tag{3}$$

The above objective deals with discrete variables and is difficult to solve. Thus the discrete constraint is relaxed and it is then solved using an alternating minimization technique [27], [16], [20]. For each variable update of H^x and H^y, we use the projected gradient descent approach [22]. Denoting the above objective as a function f with respect to a single variable, say U, where $U = \{H^x \text{ or } H^y\}$, with the other variable fixed, we need to solve the following

$$\begin{aligned} \min_{U} \quad & f(U) \\ \text{s.t.} \quad & U \in [-1, 1]^{N \times k_1}. \end{aligned} \tag{4}$$

Following [22], we compute $\frac{\partial f(U)}{\partial U}$ and then update U as $\text{Proj}_{[-1,1]}\left(U - \eta \frac{\partial f(U)}{\partial U}\right)$, where η is the step size. These steps are repeated for both H^x, H^y until convergence.

Computing the additional hash bits: Assume that we have already learned the hash bit representation of the training data as H^x, H^y of k_1 bit length. Now, in addition to the original tags $L_{tr} \in \mathcal{R}^{N \times a}$, we have access to an additional set of tags, and let the increased set be denoted as $\hat{L}_{tr} \in \mathcal{R}^{N \times \hat{a}}$ ($a \subset \hat{a}$). One can always reconstruct the new affinity as $\hat{s}_{ij} =< \hat{L}_{tr}^{i}, \hat{L}_{tr}^{j} >$ and relearn the hash bit representation $\hat{H^x}, \hat{H^y}$ from scratch. Instead, we propose to reuse the already learned hash bits H^x, H^y of k_1 bits and extend it to $\hat{H^x}, \hat{H^y}$ of k_2 bits to incorporate the additional semantic information.

This way, we can utilize the already learned bits. The additional bits are computed as:

$$\begin{aligned} &\min_{\hat{H^x},\hat{H^y}} && \|k_2\hat{S} - \hat{H^x}\hat{H^y}^T\|_F^2 \\ =&\min_{\overline{H^X},\overline{H^Y}} && \|k_2\hat{S} - (H^x H^{yT} + \overline{H^x H^y}^T)\|_F^2 \\ =&\min_{\overline{H^x},\overline{H^y}} && \|D - \overline{H^x H^y}^T\|_F^2 \end{aligned} \tag{5}$$

Here, $\overline{H^x}$ and $\overline{H^y}$ are the additional hash bits of length $(k_2 - k_1)$ to be learnt for the x and y modalities. Since $D = k_2\hat{S} - H^x H^{yT}$, we see that the additional hash bits are trying to learn the semantic information not captured by the original bit representation. The above objective (5) can be solved using a similar strategy as in (3).

A sub-problem of the one addressed above is when we want to increase the hash bit length to better capture the relationship between the data for better retrieval, even though the tags remain unchanged. This scenario is relevant since the retrieval performance of standard state-of-the-art hashing techniques [11], [20] increases with the increase in hash bits. We can use the same formulation as above to learn the additional bits and reuse the already learned k_1 bits. For this, we use (5) by replacing $\hat{S}$ with S since the tags remain the same.

3.2 Stage 2: Learning the Hash Functions

We first describe how to learn two hash functions $\mathcal{F}_x : x \rightarrow H^x$ and $\mathcal{F}_y : y \rightarrow H^y$ to generate the hash codes for the x and y modalities. Next, we will discuss how to learn the hash functions for the extended bits. Here, we consider the two modalities as image and text, but the proposed approach is equally applicable for other modalities.

Owing to the success of deep learning approaches for learning data representations, we formulate a multilayer neural network based approach to learn the hash functions. The proposed setup (Figure 3) consists of two networks, corresponding to image and text. For the image modality, we use the AlexNet [12] architecture which has convolutional (conv) layers followed by fully connected (fc) layers and replace the last fc layer with a new one of k_1 dimensions. For the text modality, the text data represented using bag-of-words is passed through a series of fc layers to learn the hash function, with the final layer being of k_1 dimensions. Additionally, we define a unification network which takes as input the concatenated output of the image and text networks and generates a unified hash code. The output of each of these three networks is passed through a tanh activation to bring the values between -1 and 1 which are finally quantized using bit-wise sign operation to get the final hash codes.

We consider each part of the network as independent modules or blocks with a specific function. For the image network, the initial part of the network up to the penultimate fc layer is defined as the Feature Block (FB_x) which generates feature vector specific to the input image. This feature vector is used by the last fc layer termed as Hashing Block (HB_x) to learn semantically rich hash codes. We define the blocks in the above manner so that more hashing blocks can be added after the feature block to increase the number of hash bits, which enables us to train only a part of the network

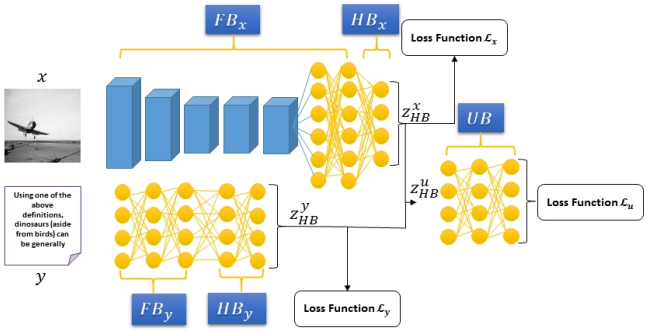

Fig. 3. The basic network architecture to learn the hash function.

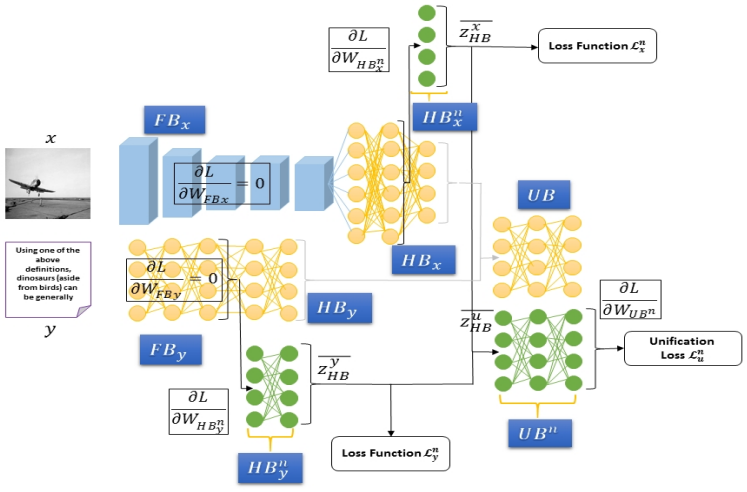

Fig. 4. The network architecture to learn the extended hash functions using the new network blocks (marked in green). The previously reused network blocks (marked in blue and yellow) are not trained again.

while obtaining better hash representations. The text domain network can be considered to have similar modules with the initial fc layers before the last two fc layers constituting the Feature Block (FB_y) and the last two fc layers making up the Hashing Block (HB_y).

We designate the functions approximated by FB and HB for the two modalities as $\{f^t_{FB}, f^t_{HB}\}$, with $t = \{x, y\}$. The functions are defined as $f^t_{FB} : t \rightarrow z^t_{FB}$ and $f^t_{HB} : z^t_{FB} \rightarrow z^t_{HB}$. Hence, $\mathcal{F}_t$ can be written as $\mathcal{F}_t = f^t_{HB}(f^t_{FB}(.))$. We concatenate the output (before the tanh layer) from the two networks f^x_{HB}, f^y_{HB} to form the concatenated vector z^u_{HB} which is then passed through the Unification Block (UB) to get

the unified hash bits. We employ mean squared error (MSE) loss at the output of the two networks to learn the hash bits H^x and H^y. MSE is also used to get the unified hash code (choosing either H^x or H^y was found to be equally good). The overall loss function is $\mathcal{L} = \mathcal{L}_x + \mathcal{L}_y + \mathcal{L}_u$ with each loss $\mathcal{L}_t, (t \in \{x, y, u\})$ defined as

$$\mathcal{L}_t = \sum_{i=1}^{N}\sum_{j=1}^{k_1}\left(z^t_{HB,ij} - h^t_{ij}\right)^2 + \lambda\sum_{i=1}^{N}\sum_{j=1}^{k_1}\left(\log\cosh(\left|z^t_{HB,ij}\right| - 1)\right) \tag{6}$$

The second term helps to reduce the quantization loss [37], with λ as the weight parameter. The quantization loss is essential because the variable h^t_{ij} (hash bits from the first stage) is discrete, whereas $z^t_{HB,ij}$ is the output after passing through the tanh layer.

Learning the additional hash functions: Now, given the extended hash codes of length $k_2(k_2 > k_1)$ bits, we show how the network described above can be seamlessly adapted to the new set of bits. In Stage 1, we keep the initially learned hash codes H^x, H^y of length k_1 fixed, and only learn the additional bits $\overline{H^x}, \overline{H^y}$ of length $(k_2 - k_1)$. Thus in Stage 2, we only introduce new hashing blocks and unification block $\{HB^n_x, HB^n_y, UB^n\}$ corresponding to the new $(k_2 - k_1)$ bits, which have the same architecture as $\{HB_x, HB_y, UB\}$ respectively. The original FB, HB and UB blocks for the initial k_1 bits need not be retrained. This is illustrated in Figure 4. The new loss is $L^n = \mathcal{L}^n_x + \mathcal{L}^n_y + \mathcal{L}^n_u$ with each loss $\mathcal{L}^n_t, (t \in \{x, y, u\})$ defined as

$$\mathcal{L}^n_t = \sum_{i=1}^{N}\sum_{j=k_1+1}^{k_2}\left(\overline{z^t_{HB^n,ij}} - \overline{h^t_{ij}}\right)^2 + \lambda\sum_{i=1}^{N}\sum_{j=k_1+1}^{k_2}\left(\log\cosh(\left|\overline{z^t_{HB^n,ij}}\right| - 1)\right) \tag{7}$$

Since we do not retrain the original blocks FB_x, FB_y, we fix $\{\frac{\partial L^n}{\partial W_{FBx}} = 0, \frac{\partial L^n}{\partial W_{FBy}} = 0\}$ (W denotes the weights and biases of the network blocks) so as not to update their network parameters, which helps to significantly reduce the training time.

There are two fc layers in the HB_y block for the text network as it is not pre-trained. The unification block needs to be repeated as the function to learn the additional bits have to be trained from scratch.

For a test query, hash codes are generated by passing it through the network and using sign() for quantization. For a paired data sample, the unified hash representation is generated by passing it further through the unification block with subsequent quantization.

This work has differences from the recent state-of-the-art technique in [20] - (1) here, we are integrating a deep based architecture to learn the hash functions which performs better than the non-deep based ones, (2) the unification scheme outperforms the unification scheme in [20] and (3) this algorithm can learn the bits in an incremental fashion. This incremental method can also be applied for unimodal hashing purpose though here we have specifically focussed on cross-modal applications.

4 Experiments

4.1 Datasets and Evaluation Protocol

The three datasets we have used for our evaluation are MIRFLICKR-25K [10], IAPR TC-12 [6] and NUS-WIDE [3]. We follow the standard experimental protocol as in [11]. MIRFLICKR-25K [10] has $25,000$ images accumulated from Flickr website with each described by textual tags, of which only those pairs with at-least 20 tags are considered. We use Convolutional Neural Network (CNN) features for images and 1386-dimensional bag-of-words (BOW) feature vectors [11] for the text. Each pair has multiple annotations out of 24 possible unique labels. The IAPR TC-12 dataset [6] has $20,000$ image-text pairs with annotation spread over 255 labels. Following the same protocol as in [11], we use the entire dataset in IAPR TC-12 for our experiments. CNN features and 2912-dimensional BOW features are used for image and text data respectively [11]. NUS-WIDE [3] has $260,648$ images designated with multiple labels across 81 categories. As in [11], we consider only those pairs that belong to the 21 most frequent concepts to form a reduced dataset of $195,834$ image-text pairs for our experiments. For textual data, we use a 1000-dimensional BOW representation, while for images we use CNN features. All the features are taken from the work in [11].

The retrieval performance is measured using Mean Average Precision (MAP). It is computed as the mean of the Average Precision (AP) of all the queries where $AP(q) = \frac{\sum_{r=1}^{R} P_q(r)\delta(r)}{\sum_{r=1}^{R} \delta(r)}$. R is the number of retrieved items and $P_q(r)$ is the precision at position r for query q. $\delta(r)$ is defined to be 1 if the r^{th} retrieved item shares at least one label with the label of the query q, else it is set to 0.

4.2 Baseline and Implementation Details

We compare the proposed approach with the state-of-the-art cross-modal supervised hashing methods like SePH [15], STMH [24], SCM [31], GSPH [20], DCMH [11], PRDH [30] and some unsupervised methods like CMFH [5] and CCA [8]. We report the results of the above baseline algorithms from the work in [11] whenever available. We report the results of GSPH and SePH using the unification strategy for fair comparison. We use the open source deep learning toolbox PyTorch [21] on a NVIDIA Titan X GPU card to perform our evaluations. For the text based network, the FB and HB consist of one and two fc layers respectively and the UB consists of three fc layers. The hidden layer is taken to be 500-dimensional except for the NUS-WIDE dataset, where it is chosen as 1500. The learning rate for training the network initially and re-training it for the extended bits is set to be $lr = 1e^{-3}$ and $lr = 1e^{-4}$. It took around $10-30$ iterations for our network to converge. Now, we describe the results obtained by our algorithm.

4.3 Results

Results for Protocol I (P-I) P-I is the standard cross-modal evaluation protocol as in [11]. We randomly sample 2000 examples to form the query set and the remaining form the retrieval set for MIRFLICKR-25K [10] and IAPR TC-12 [6] datasets. For

NUS-WIDE [3], randomly chosen 2100 examples form the query set and the remaining examples form the retrieval set. The training data is formed by sampling from the retrieval set itself as in [11]. We use $\{10000, 10000, 10500\}$ examples as the training set for MIRFLICKR-25K, IAPR TC-12, and NUS-WIDE datasets respectively. As in [11], [15], [20], a retrieval is considered correct if it shares at least one common label with the query. The MAP results for *GrowBit* and comparison with the state-of-the-art on the three datasets is given in Table 1. $I \rightarrow T$ denotes that image query is used to retrieve text data and vice-versa. We report the comparisons with PRDH [30] in Table 1 by following the same protocol as in [30] (results marked with *). We observe that *GrowBit* outperforms the other approaches for all the three datasets. The performance of our approach increases monotonically with an increase in the number of bits. Interestingly, it also outperforms DCMH [11] and PRDH [30] though ours is not an end-to-end deep learning approach. Figure 5 shows some text-image retrieval (top-5) results for the MirFlickr [10] dataset.

Table 1. MAP for MIRFLICKR-25K, IAPR TC-12 and NUS-WIDE datasets for P-I.

Task	Method	MIRFLICKR-25K			IAPR TC-12			NUS-WIDE		
		16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
$I \rightarrow T$	CCA	0.5719	0.5693	0.5672	0.3422	0.3361	0.3300	0.3604	0.3485	0.3390
	CMFH	0.6377	0.6418	0.6451	0.4189	0.4234	0.4251	0.4900	0.5053	0.5097
	SCM	0.6851	0.6921	0.7003	0.3692	0.3666	0.3802	0.5409	0.5485	0.5553
	STMH	0.6132	0.6219	0.6274	0.3775	0.4002	0.4130	0.4710	0.4864	0.4942
	SePH	0.7123	0.7194	0.7232	0.4442	0.4563	0.4639	0.6037	0.6136	0.6211
	DCMH	0.7410	0.7465	0.7485	0.4526	0.4732	0.4844	0.5903	0.6031	0.6093
	GSPH	0.7706	0.7866	0.7984	0.4799	0.5128	0.5372	0.7074	0.7258	0.7381
	GrowBit	**0.7951**	**0.8053**	**0.8232**	**0.4987**	**0.5371**	**0.5621**	**0.7069**	**0.7283**	**0.7408**
	PRDH*	0.7126	0.7128	0.7201	-	-	-	0.6348	0.6529	0.6506
	GrowBit*	**0.7675**	**0.7898**	**0.8008**	-	-	-	**0.7275**	**0.7491**	**0.7584**
$T \rightarrow I$	CCA	0.5742	0.5713	0.5691	0.3493	0.3438	0.3378	0.3614	0.3494	0.3395
	CMFH	0.6365	0.6399	0.6429	0.4168	0.4212	0.4277	0.5031	0.5187	0.5225
	SCM	0.6939	0.7012	0.7060	0.3453	0.3410	0.3470	0.5344	0.5412	0.5484
	STMH	0.6074	0.6153	0.6217	0.3687	0.3897	0.4044	0.4471	0.4677	0.4780
	SePH	0.7216	0.7261	0.7319	0.4423	0.4562	0.4648	0.5983	0.6025	0.6109
	DCMH	0.7827	0.7900	0.7932	0.5185	0.5378	0.5468	0.6389	0.6511	0.6571
	GSPH	0.7388	0.7522	0.7653	0.4881	0.5260	0.5523	0.6561	0.6716	0.6860
	GrowBit	**0.7778**	**0.7919**	**0.7994**	**0.5205**	**0.5614**	**0.5828**	**0.6595**	**0.6864**	**0.6945**
	PRDH*	0.7467	0.7540	0.7505	-	-	-	0.6808	0.6961	0.6943
	GrowBit*	**0.7496**	**0.7728**	**0.7793**	-	-	-	**0.6893**	**0.7060**	**0.7082**

Results for Protocol II (P-II) In P-II, we want to increase the length of hash bits to get better retrieval performance without changing the number of tags/labels. The results for P-II for the three datasets are given in Table 2. Here $16 \rightarrow 64$ denotes that the initial hash code length is 16 and it is increased to 64 using *GrowBit*. Comparing the results with that in Table 1, we observe that the improved performance obtained by increasing

Fig. 5. Few text-image retrieval (top-5) results for the MirFlickr dataset [10] under P-I protocol.

Table 2. MAP for the three datasets for P-II and P-III.

Protocol	Dataset	$I \rightarrow T$			$T \rightarrow I$		
		$16 \rightarrow 32$	$16 \rightarrow 64$	$32 \rightarrow 64$	$16 \rightarrow 32$	$16 \rightarrow 64$	$32 \rightarrow 64$
P-II	MIRFLICKR-25K	0.8108	0.8180	0.8253	0.7970	0.7983	0.8085
	IAPR TC-12	0.5325	0.5449	0.5599	0.5550	0.5610	0.5820
	NUS-WIDE	0.7248	0.7394	0.7413	0.6860	0.6884	0.7001
P-III	MIRFLICKR-25K	0.7907	0.8094	0.8010	0.7700	0.7777	0.7733
	IAPR TC-12	0.5143	0.5574	0.5525	0.5310	0.5647	0.5635
	NUS-WIDE	0.6933	0.7188	0.7143	0.6426	0.6540	0.6629

the number of bits eg. $16 \rightarrow 64$ and $32 \rightarrow 64$ does not significantly deviate from the expected result if we would have learned all the 64 hash bits from scratch. This is observed across both the scenarios $I \rightarrow T$ and $T \rightarrow I$ and across all the datasets. Figure 6 shows some $T \rightarrow I$ retrieval (top-5) results for the MirFlickr [10] dataset. We observe that using extended hash bits ($16 \rightarrow 64$), there is a noticeable improvement in the images that are retrieved corresponding to the text query.

Results for Protocol III (P-III) In this third and most challenging protocol, the number of tags increases which needs to be accounted for by the new bits. Here, we initially learn the hash codes using 50% of the tags and use the extended hash bits to learn the full semantic relations using all the tags. The 50% split is generated randomly and we repeat our experiments three times and report the average. The results for the three datasets for P-III are shown in Table 2. We observe that (1) the performance of the incremented hash bits as compared to Table 1 is slightly lower, and (2) the performance drop in $I \rightarrow T$ is smaller as compared to that in $T \rightarrow I$.

4.4 Analysis of the Proposed Approach

Effect of Unification: Figure 7 shows the retrieval performance (MAP) on the three datasets with and without unification for hash bit length of $16, 32$ and 64 for Protocol I. We observe that the unified bits gives a significant boost in the performance. Thus the complementary information from both the modalities results in significant improvement in the retrieval performance for both $I \rightarrow T$ and $T \rightarrow I$ scenarios.

Fig. 6. Few text-image retrieval (top-5) results for the MirFlickr [10] dataset under P-II protocol. We observe that the extended hash bits $(16 \rightarrow 64)$ helps to better capture the semantic relation and shows a noticeable improvement in the retrieved items.

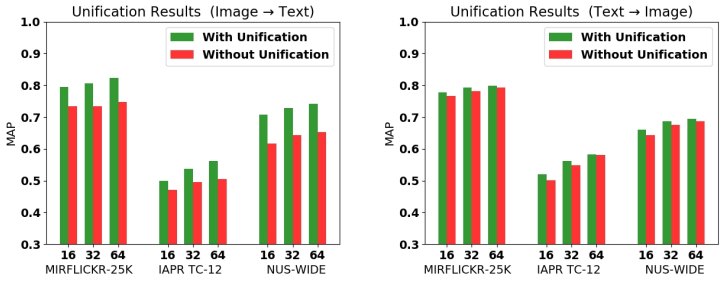

Fig. 7. MAP (y-axis) for the three datasets with and without unification for P-I.

Effect of number of hash bits: We observe from Figure 8 (left) that the retrieval performance (MAP) improves monotonically with increase in the number of bits (from 16 to 128) for both the datasets MIRFLICKR-25K and IAPR TC-12.

Effect of better network architecture: Here we study the effects of using networks with better representation capabilities on the retrieval performance. The $I \rightarrow T$ results for the MIRFLICKR-25K dataset with different architectures [21] for the image domain is shown in Figure 8 (right). We observe that better networks improve the retrieval performance for both 16 and 32 bits. Since the network for the text data remains unchanged, the $T \rightarrow I$ results do not improve much, which can potentially be improved with a better network for the text modality.

Savings in Computation: Now we show the computation savings in both Stage 1 and Stage 2 of our algorithm when we learn extended hash codes. Table 3 shows the

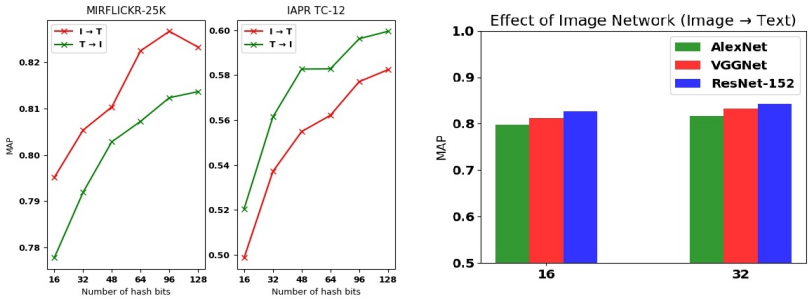

Fig. 8. (Left) MAP with increasing number of bits for both $I \rightarrow T$ and $T \rightarrow I$ scenarios under P-I protocol. (Right) MAP performance on MIRFLICKR-25K using different networks.

computation required for the MIRFLICKR-25K dataset when we extend the hash bits from $k_1 = 16$ to $k_2 = \{32, 64\}$ for a training data size of $N = 10,000$. We observe that most of the savings is obtained when the number of fc layers in the HB block is reduced. The relative number of parameters saved using GrowBit is also provided. Considerable savings will also be there during testing as k_1 bits for the test data can be reused.

Table 3. Comparison of MAP obtained for P-I and P-II. The relative gain in performance (%) obtained ($k_1 \rightarrow k_2$ vs k_1) is shown. The parameters (in millions for both Stages s_1, s_2) required for training $10K$ data of MIRFLICKR-25K is also noted with relative savings $\left(1 - \frac{\#(k_1 \rightarrow k_2)}{\#k_2}\right)$ given in %.

	$I \rightarrow T$ ($T \rightarrow I$)		
	16 bits	32 bits	64 bits
P-I	0.7951 (0.778)	0.805 (0.792)	0.823 (0.799)
P-II	-	0.8108 (0.797) $_{16 \rightarrow 32}$	0.818 (0.7983) $_{16 \rightarrow 64}$
Gain	-	1.975% (2.469%)	2.880% (2.636%)
$\#k_1$	0.32^{s_1} + 59.50^{s_2}	0.64^{s_1}+59.65^{s_2}	1.28^{s_1}+59.80^{s_2}
$\#(k_1 \rightarrow k_2)$	-	$(0.32^{s_1} + 0.59^{s_2})_{16 \rightarrow 32}$	$(0.96^{s_1} + 0.76^{s_2})_{16 \rightarrow 64}$
Parameters saved (in %)	-	$50.0\%^{s_1} + 98.9\%^{s_2}$	$25.0\%^{s_1} + 98.7\%^{s_2}$

5 Summary

In this work, we have proposed a novel incremental hashing approach *GrowBit* for cross-modal retrieval which can incrementally learn the hash codes for better representation of the data, especially when the number of tags increases. The novel hash code unification block gives significant performance boost by using the complementary information of the two modalities effectively. To the best of our knowledge, this is the first work which addresses this problem. Extensive evaluation on three cross-modal datasets under different protocols justifies the usefulness of the proposed approach.

References

1. Bronstein, M.M., Bronstein, A.M., Michel, F., Paragios, N.: Data fusion through cross-modality metric learning using similarity-sensitive hashing. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3594–3601 (2010)
2. Cakir, F., He, K., Sclaroff, S.: Hashing with binary matrix pursuit. arXiv preprint arXiv:1808.01990 (2018)
3. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: a real-world web image database from national university of singapore. In: 2009 ACM International Conference on Image and Video Retrieval (ACM-CIVR). pp. 48–56 (2009)
4. Dai, Q., Li, J., Wang, J., Jiang, Y.G.: Binary optimized hashing. In: 2016 ACM on Multimedia Conference (ACM-MM). pp. 1247–1256 (2016)
5. Ding, G., Guo, Y., Zhou, J.: Collective matrix factorization hashing for multimodal data. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2075–2082 (2014)
6. Escalante, H.J., Hernández, C.A., Gonzalez, J.A., López-López, A., Montes, M., Morales, E.F., Sucar, L.E., Villaseñor, L., Grubinger, M.: The segmented and annotated iapr tc-12 benchmark. Computer Vision and Image Understanding **114**, 419–428 (2010)
7. Fatih, C., He, K., Bargal, S.A., Sclaroff, S.: Mihash: Online hashing with mutual information. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 437–445 (2017)
8. Hardoon, D.R., Szedmak, S., Shawe-Taylor, J.: Canonical correlation analysis: An overview with application to learning methods. Neural computation **16**, 2639–2664 (2004)
9. Horn, R.A., Johnson, C.R.: Matrix analysis. Cambridge university press (1990)
10. Huiskes, M.J., Lew, M.S.: The mir flickr retrieval evaluation. In: 2008 ACM International Conference on Multimedia Information Retrieval (ACM-MIR). pp. 39–43 (2008)
11. Jiang, Q.Y., Li, W.J.: Deep cross-modal hashing. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3232–3240 (2017)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: 2012 Advances in Neural Information Processing Systems (NIPS). pp. 1097–1105 (2012)
13. Kumar, S., Udupa, R.: Learning hash functions for cross-view similarity search. In: 2011 International Joint Conference on Artificial Intelligence (IJCAI). pp. 1360–1365 (2011)
14. Lin, G., Shen, C., Suter, D., van den Hengel, A.: A general two-step approach to learning-based hashing. In: 2013 IEEE International Conference on Computer Vision (ICCV). pp. 2552–2559 (2013)
15. Lin, Z., Ding, G., Han, J., Wang, J.: Cross-view retrieval via probability-based semantics-preserving hashing. IEEE Trans. Cybern. **47**, 4342–4355 (2017)
16. Lin, Z., Ding, G., Hu, M., Wang, J.: Semantics-preserving hashing for cross-view retrieval. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3864–3872 (2015)
17. Liu, L., Lin, Z., Shao, L., Shen, F., Ding, G., Han, J.: Sequential discrete hashing for scalable cross-modality similarity retrieval. IEEE Trans. Image Process. **26**, 107–118 (2017)
18. Long, M., Cao, Y., Wang, J., Yu, P.S.: Compositional correlation quantization for large-scale multimodal search. In: 2016 International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR). pp. 579–588 (2016)
19. Mandal, D., Biswas, S.: Label consistent matrix factorization based hashing for cross-modal retrieval. In: 2017 IEEE International Conference on Image Processing (ICIP). pp. 2901–2905 (2017)
20. Mandal, D., Chaudhury, K.N., Biswas, S.: Generalized semantic preserving hashing for n-label cross-modal retrieval. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2633–2641 (2017)

21. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: 2017 Advances in Neural Information Processing Systems Workshop (NIPS-W) (2017)
22. Shen, F., Zhou, X., Yang, Y., Song, J., Shen, H.T., Tao, D.: A fast optimization method for general binary code learning. IEEE Trans. Image Process. **25**, 5610–5621 (2017)
23. Song, J., Yang, Y., Yang, Y., Huang, Z., Shen, H.T.: Inter-media hashing for large-scale retrieval from heterogeneous data sources. In: 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD). pp. 785–796 (2013)
24. Wang, D., Gao, X., Wang, X., He, L.: Semantic topic multimodal hashing for cross-media retrieval. In: 2015 International Joint Conference on Artificial Intelligence (IJCAI). pp. 3890–3896 (2015)
25. Wang, J., Kumar, S., Chang, S.F.: Sequential projection learning for hashing with compact codes. In: 2010 International Conference on Machine Learning (ICML). pp. 1127–1134 (2010)
26. Wu, B., Yang, Q., Zheng, W.S., Wang, Y., Wang, J.: Quantized correlation hashing for fast cross-modal search. In: 2015 International Joint Conference on Artificial Intelligence (IJCAI). pp. 3946–3952 (2015)
27. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: 2014 AAAI Conference on Artificial Intelligence (AAAI). pp. 2156–2162 (2014)
28. Xie, L., Shen, J., Han, J., Zhu, L., Shao, L.: Dynamic multi-view hashing for online image retrieval. In: 2017 International Joint Conference on Artificial Intelligence (IJCAI). pp. 3133–3139 (2017)
29. Xu, X., Shen, F., Yang, Y., Shen, H.T., Li, X.: Learning discriminative binary codes for large-scale cross-modal retrieval. IEEE Trans. Image Process. **26**, 2494–2507 (2017)
30. Yang, E., Deng, C., Liu, W., Liu, X., Tao, D., Gao, X.: Pairwise relationship guided deep hashing for cross-modal retrieval. In: 2017 AAAI Conference on Artificial Intelligence (AAAI). pp. 1618–1625 (2017)
31. Zhang, D., Li, W.J.: Large-scale supervised multimodal hashing with semantic correlation maximization. In: 2014 AAAI Conference on Artificial Intelligence (AAAI). pp. 2177–2183 (2014)
32. Zhang, J., Peng, Y., Yuan, M.: Unsupervised generative adversarial cross-modal hashing. In: 2018 AAAI Conference on Artificial Intelligence (AAAI). pp. 539–546 (2018)
33. Zhang, R., Lin, L., Zhang, R., Zuo, W., Zhang, L.: Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. IEEE Trans. Image Process. **24**, 4766–4779 (2015)
34. Zhang, T., Wang, J.: Collaborative quantization for cross-modal similarity search. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2036–2045 (2016)
35. Zhou, J., Ding, G., Guo, Y.: Latent semantic sparse hashing for cross-modal similarity search. In: 2014 International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR). pp. 415–424 (2014)
36. Zhou, J., Ding, G., Guo, Y., Liu, Q., Dong, X.: Kernel-based supervised hashing for cross-view similarity search. In: 2014 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6 (2014)
37. Zhu, H., Long, M., Wang, J., Cao, Y.: Deep hashing network for efficient similarity retrieval. In: 2016 AAAI Conference on Artificial Intelligence (AAAI). pp. 2415–2421 (2016)
38. Zhuang, B., Lin, G., Shen, C., Reid, I.: Fast training of triplet-based deep binary embedding networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5955–5964 (2016)