Extensible Cross-Modal Hashing

Tian-yi Chen[1] , **Lan Zhang**[1,2 *] , **Shi-cong Zhang**[1] , **Zi-long Li**[3] and **Bai-chuan Huang**[4]
[1]School of Computer Science and Technology, University of Science and Technology of China, China
[2]School of Data Science, University of Science and Technology of China, China
[3]School of Information Science and Engineering, Northeastern University, China
[4]Department of Physics, University of California Berkeley, USA
wandero@mail.ustc.edu.cn, zhanglan@ustc.edu.cn, zsc2016@mail.ustc.edu.cn,
longzili@stumail.neu.edu.cn, huangbc1998@gmail.com

Abstract

Cross-modal hashing (CMH) models are introduced to significantly reduce the cost of large-scale cross-modal data retrieval systems. In many real-world applications, however, data of new categories arrive continuously, which requires the model has good extensibility. That is the model should be updated to accommodate data of new categories but still retain good performance for the old categories with minimum computation cost. Unfortunately, existing CMH methods fail to satisfy the extensibility requirements. In this work, we propose a novel extensible cross-modal hashing (ECMH) to enable highly efficient and low-cost model extension. Our proposed ECMH has several desired features: 1) it has good forward compatibility, so there is no need to update old hash codes; 2) the ECMH model is extended to support new data categories using only new data by a well-designed "weak constraint incremental learning" algorithm, which saves up to 91% time cost comparing with retraining the model with both new and old data; 3) the extended model achieves high precision and recall on both old and new tasks. Our extensive experiments show the effectiveness of our design.

1 Introduction

With the rapid development of the Internet and smart devices, large amounts of data, such as texts, images and videos, are constantly being produced. Facing massive multi-modal data, efficient cross-modal information retrieval is badly needed in various big data applications. The main challenge of cross-modal retrieval is to solve the "media gap", since every modality has its distinct feature space. To address this issue, a popular solution is to learn a correspondence which maps data of different modalities to vector representations in an intermediate common space and expresses the similarity among these data by their distances in the common space. Traditional statistical correlation analysis based methods, e.g., Canonical Correlation Analysis (CCA), map data from different modalities to a subspace where pairwise correlations between two modalities will be maximized. Those methods, however, incur high computation and storage cost because of the high-dimensional float number vectors and the calculation of pairwise Euclidean distances. Recently, researchers introduce Cross-Modal Hashing (CMH) to significantly reduce the cost while retaining retrieval performance for large-scale cross-modal data [Bronstein *et al.*, 2010; Zhen and Yeung, 2012; Wu *et al.*, 2015; Lin *et al.*, 2015]. It maps data into a common Hamming space and uses Hamming distance instead of Euclidean distance. The common space can be learned by a Deep Neural Network (DNN), which acts as a non-linear extension of CCA, in supervised manners [Chen *et al.*, 2018; Zhang *et al.*, 2014] or unsupervised manners [Shen *et al.*, 2015; Wu *et al.*, 2018]. CMH is now widely adopted for cross-modal data retrieval [Wang *et al.*, 2016; Baltrusaitis *et al.*, 2018].

CMH models achieve good performance in cross-modal retrieval tasks; however, in many real-world applications, new data of new categories comes continuously, which requires the system has extensibility. So the model should be updated to accommodate data of new categories but still retain good performance for the old categories, and the system should still be compatible with old hash codes. Unfortunately, existing CMH models lack extensibility. Fine-tuning the model using only new data suffers from a catastrophic forgetting that results in an accuracy decrease on old tasks [Li and Hoiem, 2018]. Retraining the whole model with both old and new data incurs an ever increasing large computation cost for training and updating the hash codes for all existing data, as well as a large storage cost for all old data, There are some incremental learning methods consider the extensibility problem [Joshi and Kulkarni, 2012], but most of which focus on classification tasks [Cortes and Vapnik, 1995; Guo *et al.*, 2010]. The main techniques include adding new output dimensions for new tasks [Razavian *et al.*, 2014] and applying regularization for old tasks [Hinton *et al.*, 2015]. Those method cannot resolve the cross-modal hashing problem, since they continuously increase the dimension of the common feature space, thus the length of the hash code, whenever data of a new category comes.

In this work, we propose a novel extensible cross-modal hashing (ECMH) to empower the CMH models with good extensibility. To the best of our knowledge, this is the first work addressing the extensible cross-modal hashing problem.

*Contact Author

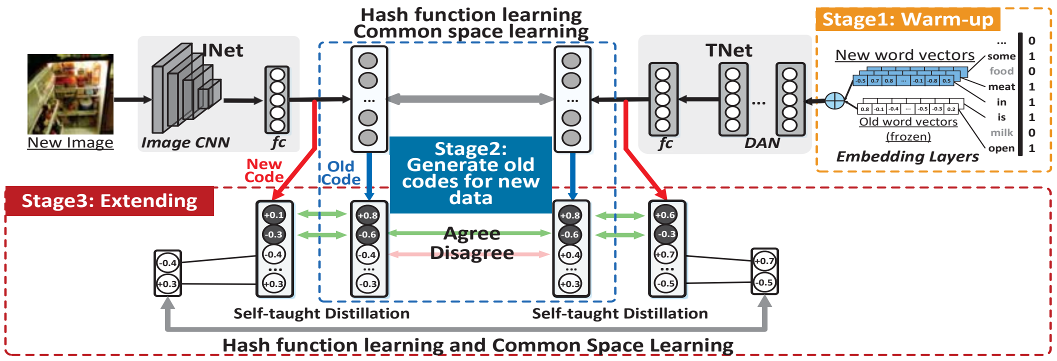


Figure 1: ECMH framework.

Our proposed ECMH has several desired features for large-scale cross-modal retrieval systems: 1) it is compatible with old hash codes, there is no need to increase the dimension of hash code or update old hash codes; 2) ECMH is extended to support new categories using only the new data by a well-designed "weak constraint incremental learning" algorithm; 3) the extended model achieves high precision and recall in both old and new tasks, and the hash codes generated by the extended model has good forward compatibility with the old hash codes. These features of ECMH enables highly efficient and low-cost cross-modal data system extension.

2 ECMH

2.1 Problem Definition

In a typical cross-modal data retrieval application, suppose we already have an old dataset with N_o cross-modal data pairs, e.g., image-text pairs, which can be represented by $\mathcal{D}_o = \{(x_o^i, y_o^i), 0 < i \leq N_o\}$ here the subscript "o" is the abbreviation for "old." With this dataset, through cross-modal hashing we can train a deep hash function $h(\mathcal{D}; \theta_o)$ with parameters θ_o for old tasks (e.g., image-text pairs of animals), that hashes two-modal data into a set of unified hash codes $\mathcal{C}_{o|\theta_o} = \{-1, +1\}^{M \times L}$. Here L is the length of the hash code, and M is the number of instances. When there comes a set of new cross-modal data pairs $\mathcal{D}_n = \{(x_n^i, y_n^i), 0 < i \leq N_n\}$ for new tasks (e.g., image-text pairs of scenes), where the subscript "n" is the abbreviation for "new", ECMH mainly focuses on the extensibility issue of the hash function. Specifically, ECMH aims to update parameters θ_o to new parameters θ_n to satisfy the following requirements: 1) we don't need to update old codes $\mathcal{C}_o$; 2) we don't need to increase the hash code length L; 3) we update the hash function $h(\mathcal{D}; \theta_o)$ using only the new dataset $\mathcal{D}_n$; 4) the hash codes $\mathcal{C}_{n|\theta_n}$ of the new dataset generated by the updated hash function $h(D; \theta_n)$ achieve good performance on both old and new tasks. These features of ECMH enables highly efficient and low-cost cross-modal data system extension with a precision guarantee.

2.2 Design Overview

As shown in Figure 1, the model of ECMH consists of two deep hashing networks for images and texts, named INet and TNet respectively. With the old dataset $\mathcal{D}_o$, we can train the model to obtain old parameters θ_o. When the new dataset $\mathcal{D}_n$ comes, the process of updating parameters to θ_n includes three stages. **Stage 1** is warm-up that initializes new word vectors of new tasks; **Stage 2** is generating codes $\mathcal{C}_{n|\theta_o}$ for the new dataset using the old parameters θ_o; **Stage 3** updates the model using the new dataset according to our proposed objective function Eq.(1).

2.3 Deep Feature Learning and Warm-up

Most existing deep hashing networks for texts support only fixed vocabulary size, which makes them incompetent to handle out-of-vocabulary (OOV) words in the newly arriving data. To address this issue, we adopt Deep Averaging Network (DAN) [Iyyer *et al.*, 2015]as TNet, and initialize new word vectors of new tasks in Stage 1. Specifically, we freeze all old parameters except those of the embedding layer of TNet and then fine-tune the embedding layer for new word vectors. Through TNet, each raw text y^i is firstly encoded into a 2000-dimensional sentence vector and then embedded into an L-dimensional hash code. INet can be any deep hashing network for images. In our implementation, INet is adapted from VGG-19 [Simonyan and Zisserman, 2015] by only modifying the number of hidden units in the last FC layers to generate L-dimensional hash codes Moreover, for both INet and TNet we apply the hyperbolic tangent function (tanh) to normalize output to $(-1, 1)$ range to accelerate the model convergence. Let $h^x(X; \theta)$ denote INet, and $h^y(Y; \theta)$ denote TNet. INet/TNet produces a deep representation for each image/text. Thus, given a cross-modal dataset, the matrices of deep representations of images and texts are F and G respectively. The generated hash code $\mathcal{C} = sign(F + G)$. Since ECMH will update the old parameters θ_o to θ_n, there will be two versions of F and G. We use a subscript $n|\theta_o$ to indicate the deep representations of data for new task generated with old parameters θ_o, and so on.

2.4 Objective Function of ECMH

It is nontrivial to accommodate the new parameters θ_n to both new and old tasks in a way satisfying the aforementioned requirements. A traditional fine-tuning method using only data for new tasks suffers the catastrophic forgetting effect, leading to deteriorating performance on old tasks. The core idea of ECMH is performing selective learning considering the cross-modal agreement of both deep representations generated with old parameters (in Step 2) and new parameters (in Step 3). Specifically, if the generated deep representations of an image-text pair have the same values on some dimensions, these "agreed" dimensions should be excluded from updating. The objective function of ECMH is designed to find "agreement" that does not need to change and keep it, meanwhile find "disagreement" and try to reach "agreement." As the following equation, the objective function is composed of three loss functions:

$$\min_{\mathcal{C},\theta_t^x,\theta_t^y} \mathcal{L} = \mathcal{L}_d + \mathcal{L}_c + \mathcal{L}_h \qquad s.t.\ \mathcal{C} \in \{-1,1\}^{M\times L} \tag{1}$$

The novel self-taught distillation loss $\mathcal{L}_d$ is proposed to keep those "agreed" dimensions unchanged according to deep representations generated with old parameters, which ensures the performance of the updated model on old tasks. The common space learning loss $\mathcal{L}_c$ and hash function learning loss $\mathcal{L}_h$ work together to minimize "disagreement" of deep representations generated with updated parameters, which improve the performance of the updated model on new tasks. In the rest of this section, we introduce the detailed design of three loss functions.

Agreement Matrix and Similarity Matrix.
First, we introduce two measurement matrices. A_α is the *Agreement Matrix* with a threshold α, which records the cross-modal agreement of $F_{n|\theta_o}$ and $G_{n|\theta_o}$, i.e., matrices of deep representations generated with old parameters. A_α could be defined as:

$$A_\alpha[i,j] = \begin{cases} 1, & sign(F_{n|\theta_o}[i,j]) = sign(G_{n|\theta_o}[i,j]), \\ & |F_{n|\theta_o}[i,j]| > \alpha, |G_{n|\theta_o}[i,j]| > \alpha \\ 0, & others \end{cases} \tag{2}$$

The threshold α ensures the agreement is not too weak since a small absolute value of the representation indicates low confidence and mutability.

The Similarity matrix S is defined as:

$$S[i,j] = \begin{cases} 1, & d_i \text{ shares at least one label with } d_j \\ 0, & d_i, d_j \text{ don't share label.} \end{cases} \tag{3}$$

where d_* represents an image or a text in the dataset.

Self-taught Distillation Loss.
The novel self-taught distillation loss is to maintain the performance of the updated model on old tasks. The underlying assumption is that if the output with θ_n and θ_o follow a similar probability distribution, the models with θ_n and θ_o have similar performance. Given a cross-modal data pair, the agreement matrix A_α acts as a selector that samples the dimensions which should be unchanged within its output for this input data, while $(\mathbf{1} - A_\alpha)$ samples the dimensions that should be changed. Then utilizing the old representations $F_{n|\theta_o}, G_{n|\theta_o}$, the self-taught distillation loss is defined as

$$\begin{aligned}\mathcal{L}_d =& \lambda_d(\| A_\alpha \circ (F_{n|\theta_n} - F_{n|\theta_o})\|_F^2 \\ &+ \| A_\alpha \circ (G_{n|\theta_n} - G_{n|\theta_o})\|_F^2).\end{aligned} \tag{4}$$

$\|\cdot\|_F$ is the Frobenius norm of a matrix, and $\circ$ is the Hadamard product. Suppose $F_n \in \mathbb{R}^{M\times L}$, then $\lambda_d = \frac{1}{M\times L^2}$ serves as the scaling factor.

Common Space Learning.
The common space learning loss $\mathcal{L}_c$ is utilized to construct the common space by forcing the model to generate similar hash codes for semantically similar instances and vice versa. The similarity of instances is defined as Eq. (3), and the similarity of deep representations is defined as:

$$\Delta = ((\mathbf{1} - A_\alpha) \circ F_{n|\theta_n})G_{n|\theta_n}^T \qquad s.t. F_{n|\theta_n}, G_{n|\theta_n} \in \mathbb{R}^{M\times L} \tag{5}$$

As mentioned above, $(\mathbf{1} - A_\alpha)$ samples the disagreed dimensions and then we can calculate the similarity of deep representations on these disagreed dimensions. We define $\|\cdot\|_S$ to represent the sum of all elements in a matrix for simplicity. By minimizing the term $\|(\mathbf{1} - S) \circ \Delta\|_S$, we can reduce the similarity between dissimilar instances. Minimizing the term $\|S \circ (log(1 + e^\Delta) - \Delta)\|_S$ maximizes the similarity between similar instances. Combining both terms as below, we get the loss function for common space learning:

$$\mathcal{L}_c = \lambda_c(\|(1 - S) \circ \Delta\|_S + \|S \circ (log(1 + e^\Delta) - \Delta)\|_S). \tag{6}$$

Here the scaling factor λ_c is $\frac{1}{M^2\times L}$. There could be other possible choices of L_c, as long as they obey the core idea.

Hash Functions Learning.
The hash function learning loss is defined as:

$$\begin{aligned}\mathcal{L}_h =& \lambda_h(\|(1-A)\circ(\beta\mathcal{C} - F_{n|\theta_n})\|_F^2 \\ &+ \|(1-A)\circ(\beta\mathcal{C} - G_{n|\theta_n})\|_F^2)\end{aligned} \tag{7}$$

The scaling factor λ_h is $\frac{1}{M^2\times L}$. This loss is similar to that of previous work like DCMH [Jiang and Li, 2017] and SSAH [Li *et al.*, 2018], except for the hash code smoothing factor β. Since the output of TNet and INet are rescaled to the range $(-1, 1)$ by tanh, the $\mathcal{C} \in \{-1, 1\}^{M\times L}$ will be too intense to be the targeted hash codes. In practice, we observe that this issue results in a great disparity between the average of $F_{n|\theta_n}$ and $G_{n|\theta_n}$. Some efforts assign weights to each modality and partially solve the problem. Rather than tweaking the weights, our solution smoothes the targeted hash code $\mathcal{C}$ with β to reduce the difference significantly. This design helps prevent overfitting when training the old model.

2.5 Optimization

The optimization process of ECMH including the following steps: training old model for old tasks, warm up the TNet for new vocabulary and extending the model for new tasks. Like most previous solutions, we adopt an alternating strategy in every step to solve the non-convex objective function iteratively.

As presented in Algorithm 1, the core algorithm is designed to train the model according to the objective function defined in Eq. 1. Firstly, we fix parameter θ^y of TNet and hash codes $\mathcal{C}$. A mini-batch of images is sampled from the whole train set to update θ^x. Then θ^x and $\mathcal{C}$ are fixed. A

Algorithm 1 Core Algorithm

```
Require: Image set $X$, text set $Y$, similarity matrix $S$, agreement matrix $A_\alpha$, old representations $F_{n|\theta_o}, G_{n|\theta_o}$, old parameters $\theta_o^x, \theta_o^y$, mini-batch size $m$, learning rate $\mu$.
Ensure: Parameters $\theta_n^x, \theta_n^y$ of INet and TNet.
 1: Initialize: iter ← $\lceil Sizeof(X)/m \rceil$, $(\theta_n^x, \theta_n^y) \leftarrow (\theta_o^x, \theta_o^y)$
 2: repeat
 3:     for 1, ..., iter do
 4:         Sample a mini-batch $\mathcal{M}$ with $m$ images from $X$.
 5:         F ← $h^x(\mathcal{M}; \theta_n^x)$
 6:         Update $\theta_n^x$ by SGD with BP:
 7:         $\theta_n^x \leftarrow \theta_n^x - \mu\nabla_{\theta_n^x}(\mathcal{L}_d(F_{n|\theta_n}, F_{n|\theta_o}) + \mathcal{L}_c(F_{n|\theta_n}, G_{n|\theta_n}, S) + \mathcal{L}_h(F_{n|\theta_n}, \mathcal{C}))$
 8:     end for
 9:     for 1, ..., iter do
10:         Sample a mini-batch $\mathcal{M}$ with $m$ images from $Y$.
11:         G ← $h^y(\mathcal{M}; \theta_n^y)$
12:         Update $\theta_n^y$ by SGD with BP:
13:         $\theta_n^y \leftarrow \theta_n^y - \mu\nabla_{\theta_n^y}(\mathcal{L}_d(G_{n|\theta_n}, G_{n|\theta_o}) + \mathcal{L}_c(G_{n|\theta_n}, F_{n|\theta_n}, S) + \mathcal{L}_h(G_{n|\theta_n}, \mathcal{C}))$
14:     end for
15:     Update $\mathcal{C}$ by Eq. (8).
16: until Convergence
```

Algorithm 2 Extending Model for New Tasks

```
Require: Image set $X_n$ and text set $Y_n$ for new tasks, similarity Matrix $S$, mini-batch size $m$, learning rate $\mu$, parameters $\theta_o^x, \theta_o^y$ of the to-be-extended model.
Ensure: Parameters $\theta_n^x, \theta_n^y$ of INet and TNet.
 1: Initialize: iter ← $\lceil Sizeof(X)/m \rceil$, $(\theta_n^x, \theta_n^y) \leftarrow (\theta_o^x, \theta_o^y)$
 2: Freeze $\theta_n^x$,$\theta_n^y$ except for the embedding layer.    ▷ Warm-up
 3: $F_{n|\theta_o} \leftarrow h^x(X_n; \theta_o^x)$
 4: Learn $\theta_n^y$ by the Core Algorithm.
 5: Unfreeze $\theta_n^x, \theta_n^y$ except for CNN layers.    ▷ Extending
 6: Calculate $A_\alpha$ by the Eq. (2).
 7: $G_{n|\theta_o} \leftarrow h^y(Y_n; \theta_o^y)$
 8: Learn $\theta_n^x, \theta_n^y$ by the Core Algorithm.
```

mini-batch of texts is sampled to update θ^y. These first two steps utilize the stochastic gradient descent (SGD) [Bottou, 2010] with the back-propagation (BP) algorithm to update the network parameters. Finally, we fix θ^x and θ^y and update $\mathcal{C}$ through the following equation:

$$\mathcal{C} = sign(F + G) \tag{8}$$

Leveraging the core algorithm as the main component, we train the model for old tasks and extend the model for new tasks as follows. We can easily adapt the core algorithm to train the model for old tasks by setting A_α to an all-zero matrix since we have no pre-knowledge of the agreement matrix and no dimension needs to stay unchanged when training a model from scratch. In extending a model, the first thing is to ”warm-up” the new word vectors on the new tasks. We freeze all parameters but the embedding layer in TNet and train the network using the core algorithm. Then, we unfreeze all FC layers and start extending. The process is in Algorithm 2.

3 Experiment

3.1 Datasets and Experiment Setting

In our experiments, we adopt two popular datasets with image-text pairs. The **MIRFLICKR-25k** dataset [Huiskes and Lew, 2008] contains 25,000 image-text pairs collected from Flickr. For a fair comparison, we reduce the number of instances to 20,015 following the experiment protocols given in DCMH [Jiang and Li, 2017]. Each text is represented by a 1386-dimensional BoW vector and the corresponding image is rescaled to a $(224, 224, 3)$ RGB tensor. We also adopt the **MSCOCO-2014** dataset [Lin *et al.*, 2014] for its high-quality annotations. There are 55% of the images in MSCOCO labeled with “person”, which makes the new tasks too similar to the original ones. In this case, all methods perform well when extending on the original MSCOCO dataset. To evaluate our design in more general scenarios, we remove all “person” images and obtain a dataset with 36,869 instances. Each image is rescaled to $(224, 224, 3)$ and annotated with at most ten words. Unlike some previous work, we conduct the evaluation in a more stringent condition that the validation set does not contain any training data. In experiments using MIRFLICKR-25k, 10,015 instances are randomly chosen as the train set, and the rest 10000 are used for validation, namely 2000 for the query and 8000 for the database. In experiments using MSCOCO, 16,869 randomly chosen instances are used for training, and the rest 5000 and 15000 instances are used as query and database, respectively.

For old tasks and new tasks, further we split each training/validation dataset into two parts by categories of labels. There are many ways to divide categories into old and new ones. Without loss of generality, we consider two extension cases.

Super-Category Extension. Super-categories, such as “Animal” and “Vehicle”, contain quite different concepts, while sub-categories, such as “Dog”, “Cat”, “Bike” and “Car”, may have similar concepts, which makes extension for new super-categories is more challenging. In MSCOCO, we divide data by their super-categories and use about 20,000 instances of super-categories“Animal”, “Appliance”, and “Indoor” as new tasks to extend the model trained by the data of the rest super-categories. Following the definition in MSCOCO, in Flickr25k, ”Animal, People, Plant, Water, Transport, Sky, Food” are super-categories, and the rest are sub-categories. Similarly, we use about 8,000 instances of super-categories “People” and “sky” as new tasks.

Sub-Category Extension. In Flickr25k, we randomly select 6 sub-categories with 8,200 instances as new tasks. In MSCOCO, 16 sub-categories with 12000 instances are selected as new tasks. All images of new categories will only be used in the extending stage to make sure that they are totally new (sharing zero label with the old training images) for the model.

Implementation Details. We implement ECMH via Pytorch. We set all learning rate to 1.5 and decrease it by 5% every 100 steps. α is set to the range of $[0.1, 0.15]$ and β is set to 0.5. Batch size is fixed to 500. All experiments are conducted on a server with 4 TITAN X GPUs.

		Flickr_Sub		Flickr_Super		COCO_Sub		COCO_Super	
		$T \rightarrow I$	$I \rightarrow T$	$T \rightarrow I$	$I \rightarrow T$	$T \rightarrow I$	$I \rightarrow T$	$T \rightarrow I$	$I \rightarrow T$
Retrieve Old Codes	Old Model	0.7366	**0.7130**	**0.7605**	**0.7296**	0.4101	0.4099	0.4777	0.4743
	Fine-tuning	0.7114	0.6918	0.6960	0.6699	0.3967	0.4055	0.4312	0.4650
	ECMH	**0.7394**	0.7058	0.7557	0.7181	**0.4178**	**0.4342**	**0.4916**	**0.4789**
Old Tasks	Joint-DCMH	0.6543	0.6644	0.6434	0.6558	0.4004	0.3067	0.4134	0.3155
	Joint-SSAH	0.6597	0.6896	0.6579	0.6842	0.4333	0.3810	0.4160	0.3835
	Joint-ECMH	**0.6712**	**0.7145**	**0.6593**	**0.7043**	**0.4898**	**0.4052**	**0.4808**	**0.4263**
	Fine-tuning	0.6284	0.6731	0.5729	0.5984	0.4618	0.3707	0.4002	0.3403
	ECMH	**0.6557**	**0.7002**	**0.6279**	**0.6763**	**0.4930**	**0.4155**	**0.4450**	**0.4426**
New Tasks	Joint-DCMH	0.7899	0.7384	0.8267	0.7556	0.4016	0.3664	0.3433	0.3328
	Joint-SSAH	0.7738	0.7387	0.7837	0.7495	0.4302	0.4295	0.3923	0.4128
	Joint-ECMH	**0.8051**	**0.7707**	**0.8369**	**0.7922**	**0.4907**	**0.4896**	**0.4402**	**0.4522**
	Fine-tuning	**0.8067**	0.7416	**0.8201**	0.6781	**0.5136**	**0.5228**	**0.4426**	**0.4483**
	ECMH	0.7958	**0.7464**	0.7839	**0.7190**	0.4948	0.4951	0.4221	0.4427
All Tasks	Joint DCMH	0.7414	0.6947	0.7418	0.6939	0.3358	0.3267	0.3363	0.3300
	Joint SSAH	0.7322	0.7097	0.7318	0.7091	0.3836	0.3973	0.3858	0.3994
	Joint-ECMH	**0.7737**	**0.7375**	**0.7736**	**0.7378**	**0.4274**	**0.4335**	**0.4356**	**0.4403**
	Fine-tuning	0.7416	0.6975	0.6697	0.6379	0.4165	0.4217	0.3927	0.3955
	ECMH	**0.7569**	**0.7191**	**0.7336**	**0.6926**	**0.4437**	**0.4422**	**0.4223**	**0.4334**

Table 1: MAP of different methods on different tasks. The best MAP of each task is highlighted in bold.

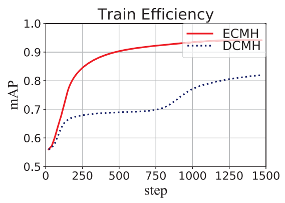


Figure 3: MAP during training process.

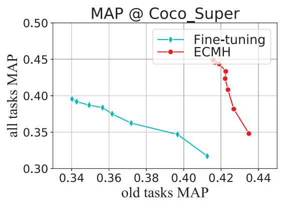


Figure 4: MAP of ECMH and Fine-tuning.

	JointECMH	ECMH_old	ECMH_new
COCO_Super	5507	737	1833
COCO_Sub	5117	1010	912
Flickr_Super	3416	731	312
Flickr_Sub	3605	727	1229

Table 2: Training runtime (sec) for JointECMH and ECMH for old tasks and new tasks.

3.2 Methods for Comparison

Since there is no existing work directly addressing the extensible cross-modal hashing issue, we compare our proposed ECMH with two most relevant state-of-the-art cross-modal hashing methods DCMH [Jiang and Li, 2017] and SSAH [Li *et al.*, 2018] by extending them in a **joint-training** manner. That is, when new data comes, we train ECMH, DCMH and SSAH using both old and new data and refer to these models as Joint-ECMH, Joint-DCMH and Joint-SSAH respectively. They provide the upper bound of precision and recall in all tasks. We also compare ECMH with a model trained in a traditional **fine-tuning** way. That is we first train an ECMH model using old data then directly fine-tune parameters using new data without considering agreement of old codes. All image networks utilize CNN layers of VGG-19 pretrained on ImageNet dataset. We use identical training set and validation set for all these methods and report their best results.

In the following experiments, we use the Mean Average Precision (MAP) and the precision-recall (PR) curve to evaluate different methods.

3.3 Training Efficiency

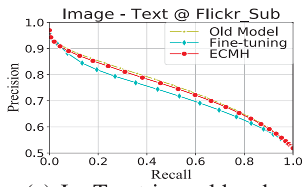
Figure 3 shows the MAP variation during the training phase of ECMH and DCMH. Due to the tanh activation function which rescales the output to $(-1, 1)$ and the light-head TNet, our proposed ECMH achieves an about three times faster convergent speed than DCMH. We further compare the time consumption of ECMH and JointECMH, as shown in Table 2, ECMH significantly reduces time cost for the model extension for new tasks. As an example, in Super-Category Extension case, ECMH saves about 91% runtime to extend the model for new tasks using Flickr dataset and saves about 67% runtime using the MSCOCO dataset.

3.4 Performance on Different Tasks

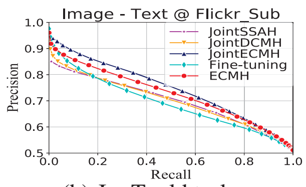
We illustrate MAP of different methods in all cases in Table 1 and PR curves using the Flickr dataset in Figure 2. We omit the PR curves using the MSCOCO dataset due to space limitation, whose performance is similar to that of the Flickr dataset. "T→I" indicates "using texts to query images" and so on.

Forward Compatibility. The forward compatibility is measured by the effectiveness of retrieving old hash codes generated by old models using new hash codes generated by new models. As shown in Table 1, on the Flickr25k dataset, thanks to the design of our loss functions, ECMH achieves almost the same MAP (with a less than 1.1% decrease) as that of directly using the old model. Fine-tuning, however, suffers from a much larger up to 6.5% decrease. Surprisedly, on the MSCOCO dataset, our method achieves even better MAP (up to 2.4% increase) than the old model while Fine-tuning still has a significant up to 11.3% decrease. The first column in Figure 2 also presents that our method has similar PR curves as the old model and significantly outperforms Fine-tuning. The results reveal that our method can not only provide good forward computability of hash codes, but also utilize new data to further improve the performance on old tasks in some cases.

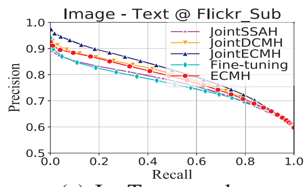
Current Performance on Old Tasks. In this evaluation, we use old data as query data and combine old and new data



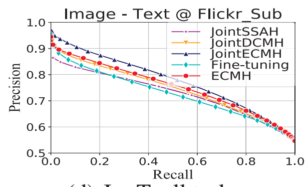
(a) I→T: retrieve old code.



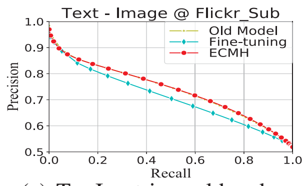
(b) I→T: old tasks.



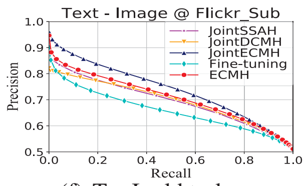
(c) I→T: new tasks.



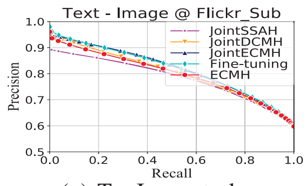
(d) I→T: all tasks.



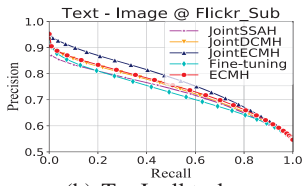
(e) T→I: retrieve old code.



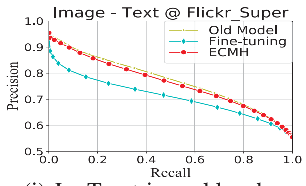
(f) T→I: old tasks.



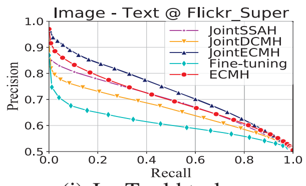
(g) T→I: new tasks



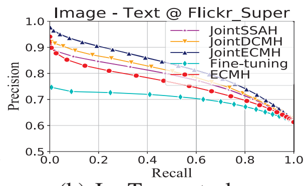
(h) T→I: all tasks.



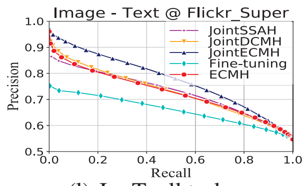
(i) I→T: retrieve old code.



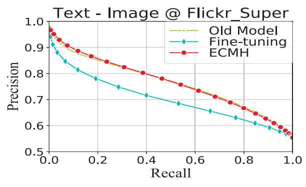
(j) I→T: old tasks.



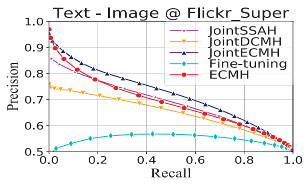
(k) I→T: new tasks.



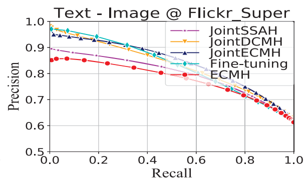
(l) I→T: all tasks.



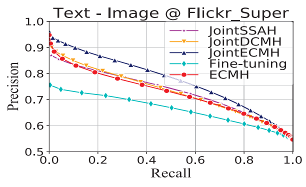
(m) T→I: retrieve old code.



(n) T→I: old tasks.



(o) T→I: new tasks.



(p) T→I: all tasks.

Figure 2: Precision-recall curve of different methods on different tasks using the Flickr dataset. Figure (a) to Figure (h) are evaluated in the Sub-Category Extension case, and Figure (i) to Figure (p) are evaluated in the Super-Category Extension case. The code length is 64.

as the database. Table 1 shows that, on the whole, three joint-training models using both old and new data have better MAP than ECMH and Fine-tuning using only new data. Among three joint-training models, joint-ECMH achieves significant better MAP than joint-DCMH and joint-SSAH in all cases. When extending models using only new data, ECMH outperforms Fine-tuning in all cases. In the worst case, i.e., Super-Category Extension on Flickr25k, Fine-Tuning has an 11.0% MAP decrease comparing with Joint-ECMH, while ECMH only has a 3.0% decrease and still achieves comparable performance to other joint training methods. The first column in Figure 2 further illustrates that in the worst case, ECMH has a much better PR curve than Fine-tuning. The result reveals that our method significantly relieves the catastrophic forgetting on old tasks and gets better old-task performance.

Current Performance on New Tasks. In this evaluation, we use new data as query data and combine new and old data as the database. The MAP results are in the third row (*new tasks*) in Table 1 and PR curves are shown in the third column in Figure 2. Since Fine-tuning update models only for new tasks, as expected, it achieves the best performance in many cases (see both Table 1 and Figure 2). ECMH has a comparable performance to Fine-tuning with an up to 2.7% decrease on the MSCOCO dataset. On the Flickr25k dataset, ECMH even has better performance on the image-query-text tasks.

The Overall Performance. Combining both new data and old data as the query data, we evaluate the overall performance of the new model. Both Table 1 and Figure 2 proves that, among three joint-training models, joint-ECMH achieves significant better MAP than joint-DCMH and joint-SSAH in all cases; and when using only new data, ECMH outperforms Fine-tuning in all cases. In the worst case (Super-Category Extension on Flickr25k), ECMH has a 6.1% MAP decrease than joint-ECMH, while Fine-tuning has a 13.5% MAP decrease. In Sub-Category Extension on MSCOCO, ECMH performs even better than all joint-training methods. This result indicates that although our self-taught distillation slightly limits the performance on new tasks, it preserves the most crucial knowledge which renders it possible to get a model as good as a jointly trained one.

Acknowledgments

This work is supported by the National Key R&D Program of China 2017YFB1003003, NSF China under Grants No. 61822209, 61751211, 61572281, 61520106007, and the Fundamental Research Funds for the Central Universities.

References

[Baltrusaitis *et al.*, 2018] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:423–443, 2018.

[Bottou, 2010] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.

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

[Chen *et al.*, 2018] Zhen-Duo Chen, Wan-Jin Yu, Chuan-Xiang Li, Liqiang Nie, and Xin-Shun Xu. Dual deep neural networks cross-modal hashing. In *AAAI Conference on Artificial Intelligence*, pages 274–281, 2018.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[Guo *et al.*, 2010] Gongde Guo, J Huang, and Lifei Chen. Knn model based incremental learning algorithm. *Pattern Recognition and Artificial Intell.*, 23:701–707, 10 2010.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *arXiv: Machine Learning*, 2015.

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

[Iyyer *et al.*, 2015] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691. Association for Computational Linguistics, July 2015.

[Jiang and Li, 2017] Q. Jiang and W. Li. Deep cross-modal hashing. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3270–3278, July 2017.

[Joshi and Kulkarni, 2012] Prachi Joshi and Parag Kulkarni. Incremental learning: Areas and methods-a survey. volume 2, page 43. Academy & Industry Research Collaboration Center (AIRCC), 2012.

[Li and Hoiem, 2018] Zhizhong Li and Derek Hoiem. Learning without forgetting. pages 614–629, 2018.

[Li *et al.*, 2018] Chao Li, Cheng Deng, Ning Li, Wei Liu, Xinbo Gao, and Dacheng Tao. Self-supervised adversarial hashing networks for cross-modal retrieval. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4242–4251, 2018.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[Lin *et al.*, 2015] K. Lin, H. Yang, J. Hsiao, and C. Chen. Deep learning of binary hash codes for fast image retrieval. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 27–35, June 2015.

[Razavian *et al.*, 2014] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 512–519, June 2014.

[Shen *et al.*, 2015] Fumin Shen, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, Zhenmin Tang, and Heng Tao Shen. Hashing on nonlinear manifolds. *IEEE Transactions on Image Processing*, 24:1839–1851, 2015.

[Simonyan and Zisserman, 2015] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *international conference on learning representations*, 2015.

[Wang *et al.*, 2016] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data a survey. *arXiv: Learning*, 104:34–57, 2016.

[Wu *et al.*, 2015] Botong Wu, Qiang Yang, Wei-Shi Zheng, Yizhou Wang, and Jingdong Wang. Quantized correlation hashing for fast cross-modal search. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3946–3952. AAAI Press, 2015.

[Wu *et al.*, 2018] Gengshen Wu, Zijia Lin, Jungong Han, Li Liu, Guiguang Ding, Baochang Zhang, and Jialie Shen. Unsupervised deep hashing via binary latent factor models for large-scale cross-modal retrieval. In *IJCAI*, pages 2854–2860, 7 2018.

[Zhang *et al.*, 2014] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. Supervised hashing with latent factor models. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 173–182. ACM, 2014.

[Zhen and Yeung, 2012] Yi Zhen and Dit-Yan Yeung. A probabilistic model for multimodal hash function learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 940–948. ACM, 2012.