Deep Hashing for Semi-supervised Content Based Image Retrieval

Muhammad Khawar Bashir[1,2] and Yasir Saleem[1]
[1]Department of Computer Science & Engineering, University of Engineering and Technology, Lahore, Pakistan
[e-mail: mkbashir@uvas.edu.pk]
[2]Department of Statistics & Computer Science, University of Veterinary & Animal Sciences, Lahore, Pakistan
*Corresponding Author: Muhammad Khawar Bashir

Received October 23, 2017; accepted April 2, 2018; published August 31, 2018

Abstract

Content-based image retrieval is an approach used to query images based on their semantics. Semantic based retrieval has its application in all fields including medicine, space, computing etc. Semantically generated binary hash codes can improve content-based image retrieval. These semantic labels / binary hash codes can be generated from unlabeled data using convolutional autoencoders. Proposed approach uses semi-supervised deep hashing with semantic learning and binary code generation by minimizing the objective function. Convolutional autoencoders are basis to extract semantic features due to its property of image generation from low level semantic representations. These representations of images are more effective than simple feature extraction and can preserve better semantic information. Proposed activation and loss functions helped to minimize classification error and produce better hash codes. Most widely used datasets have been used for verification of this approach that outperforms the existing methods.

Keywords: Semi-supervised deep hashing, binary hashing, convolutional auto-encoders, deep learning, image semantics

1. INTRODUCTION

Content based image retrieval from multimedia data that includes images and videos is based on semantic search. Hashing techniques have been presented in many applications [1]–[10] such as image retrieval because they can generate binary codes based on contents. Dimensionality reduction is the key principle in these techniques that create binary code (low dimensional data) from image visual data (high dimensional data). There are some supervised and some un-supervised learning techniques in the literature that mentioned hashing methods [6], [7], [11], [12] to extract features from raw input image data and generate hash codes. Supervised techniques use labeled data to construct hash code that can learn better by capturing the semantics of data structure. While unsupervised learning methods use unlabeled data to group images with similar semantics and generate binary code. Recently some deep hashing techniques have also been proposed by [7], [10], [13], [14]. These techniques applied deep learning methods to extract hashing codes.

Image representations are very important for accurate semantic representation and CNNs (Convolutional Neural Networks) works for it. These mid-level representations , known as deep features, can also be helpful for classification and object detections [15]–[20]. These extracted features showed better performance than handcrafted features as GIST [21] and HOG [22]. Problem with these networks is that they need huge dataset with multiple categories for training. Pre-trained CNNs are available for further fine-tuning and transfer learning. Same pre-trained networks are also capable of providing features that can be used to generate binary codes on semantic basis [23].

Deep features from unlabeled data can be extracted using convolutional autoencoders. Such features are good candidates for unsupervised learning [24] as these are generated from original images during encoding process and images are recreated using these deep features. Autoencoders have fully connected layers with convolutional layers [25]. J. Zhao et al . [26] and A. Makhzani et al. [27] proposed deep autoencoders for deep hashing with the objective to learn hierarchical representations.

Images are represented by hand-crafted features in most of the supervised and unsupervised hashing methods that cannot truly represent the semantic information. Due to real contributions of deep networks for feature extraction in image classification [7], [10], [13, p.], [14] has proposed some deep hashing methods. [8] used a method based on two steps, hash codes are learnt in first step while second step generate hash function and image representations using Convolutional Neural Network Hashing (CNNH). Network in Network Hashing (NINH) proposed by [7] overcome the problems of CNNH by using single step instead of two. Similar types of hashing methods have been used in [9], [13]. Aforementioned methods are supervised one that depend on labels generated by humans that is not good approach for real time applications.

In proposed method deep learning has been used to extract features with deep convolutional autoencoders (CAE) and hashing technique to generate different bits' hash codes. In this method CAE is involved to extract features from middle layer of encoding and decoding procedures. Activation function of sgn(.) transfers deep features to binary codes. Semi-supervised loss functions are helpful to preserve the semantic similarity and to detect multi objects in an image. Extensive experimentation has been performed on different benchmark datasets that produced favorable results. In remaining paper, Section II presents review of literature with some hashing and features extraction techniques. Section III

presents our methodology in detail with activation and loss functions. Section IV contains experiments details and Section V concludes this study.

2. Related Work

Learning based algorithms are most popular to construct hash codes. Deep networks have made considerable progress in recent years to learn hash function. Deep networks are using major three categories for learning: supervised, unsupervised and semi-supervised. Our work is using semi-supervised method to accomplish image recognition using image contents.

2.1 Learning based Algorithms for Hashing

Supervised learning is training any machine learning algorithm for input with corresponding output that, later on, will be able to identify target for any new input. Different hashing approaches that used supervised learning includes binary reconstructive embedding (BRE) [28], minimal loss hashing (MLH) [29], supervised discrete hashing (SDH) [30], supervised deep hashing (SDH) [31], deep multi-view hashing (DMVH) [32] and many more.

Unsupervised learning algorithms only use set of inputs for training and find a structure or relationship between these inputs that leads to clustering. Some hashing techniques using unsupervised learning are heterogeneous deep hashing (HetDH) [33], kernelized local sensitive hashing (KLSH) [5], spectral hashing (SH) [6] and iterative quantization (ITQ) [1].

Semi-supervised algorithms train from both labeled and unlabeled inputs to learn hash functions. Semi-supervised hashing (SSH)[34] used labeled data for semantic similarity. Formula used in SSH has two part: first part tries to minimize error for labeled data while second part, that is basically unsupervised part, tries to maximize the objective. Semi-supervised tag hashing (SSTH) [35] performed supervised learning with class labels and hash codes while unsupervised learning between input images.

In [3] Kan et. al. proposed SKHL algorithm for semi-supervised learning that used kernel hyperplane hashing function to extract features from unlabeled data. Further these features are combined with weakly labeled side information to generate parameterized hashing code that is further maximized using Fisher like objective function to learn better hashing functions. In most of the methods deep networks are used to extract features and then further used to calculate hash using some hashing function but question is “Do these extracted features fully represent the original input image?”. To solve this question, we opted autoencoders.

Autoencoders were introduced by Hinton [36] to learn features as a whole vector of parameters and conclude that these parameters are way better than hand crafted features and current employed neural networks with variations in position, orientation, scale and lighting as the parameters can recreate original image.

2.2 Framework for Generalized Autoencoder:

To explain a general autoencoder framework we can use NPN autoencoder where n and p are positive integers and we consider it $0 < p < n$ as shown in **Fig. 1**.

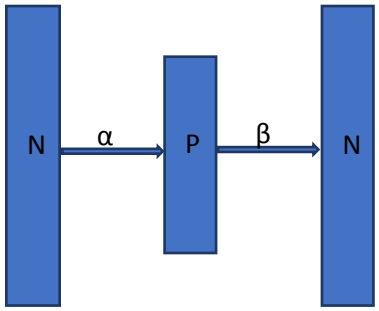

Fig. 1. Generalized Autoencoder

α is a function that generate feature parameters from input with n nodes while β is used as reverse process to generate original image. α is known as encoding while β is decoding process.

3. Semi-supervised Convolutional Autoencoders for Deep Hashing

For a given set of *N* images *X* with labels *Y*, (*X*n, *Y*n) = {(x1, y1), (x2, y2), ……., (xn, yn)}. Our goal is to learn a mapping function *Z*: *X* → {0,1}*Q* that can encodes any image x ϵ X into Q-bit binary code by preserving semantic similarity. Convolutional autoencoder was used by Ranzato et. al. [37] with four hidden units to learn non-invariant features from given output. It was unsupervised method that detects sparse features' structure invariant to small distortions.

In this study we proposed DSH, a combination of convolutional autoencoder with deep hashing to convert an image to binary code by preserving semantic similarity and image structure at the same time. Proposed method has three main parts: features extractor, hashing function, loss function. Feature extraction is accomplished using convolutional autoencoder as convolution is most commonly known technique to extract features for image recognition and autoencoders ensure that these are the features that can be used to revert back to original image and stores semantic information along with image structure. Hashing function works as layer in this network and transform an image to Q-bit binary hash code. Lastly loss function that ensures the network is preserving the semantic similarity and image structure for accurate hashing. This goal can be achieved by minimizing the empirical and embedding error with labeled and unlabeled data. These three parts work together as a network and perform image representation as a learnt hash binary code.

3.1 Feature extraction using CAE:

We choose convolutional autoencoder that consists of convolutional and pooling layers. Structure of autoencoder is given in **Fig. 2**.

3.2 Encoding:

Five groups of convolutional layers have been used along with max-pooling layer after each convolutional layer to encode an image. This structure is similar to [17] that used 64, 128, 256, 512, 512 filter respectively in convolutional layers. Two fully connected layers have been used at the end of encoding.

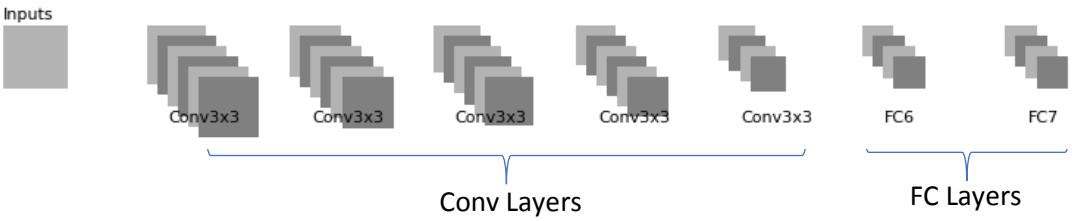

Fig. 2. Convolutional Autoencoder Model for encoding and features extraction

3.3 Hashing

Binary hash codes are learned from the last fully connected layer of encoding part. To calculate hash code, we introduced hidden layer as bcoding H with Q units right after FC7. Let WH ϵ $\mathbb{R}^{d \times Q}$ be the weights after FC7 layer of encoding function. For any image I_n with feature vector $a_n^7 \in \mathbb{R}^d$ in layer FC7, activations of layer H can be calculated as $a_n^H = \sigma(a_n^7 W^H + b^H)$, where a_n^H is a Q-dimensional vector, b^H is bias term and $\sigma(.)$ is logistic sigmoid function. Using sgn(.) for binarization will lead to non-smooth and non-convex output as its gradient will be zero for all non-zero input and not properly defined for zeros. With this type of output deep networks will not be feasible for training as it'll lead to vanishing gradient problem. This problem can be solved by smoothing the original function and making it a different and easier to optimize problem. Gradually smoothing will be reduced during training that will results in sequence of optimization problems and converging to the original optimization problem. We can make a smooth objective function motivated by the continuation methods that can converge to the desired objective. Tanh is used to make the smooth the objective function as it is similar to sigmoid and nonlinear in nature.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1} = 1 - \frac{2}{e^{2x}+1} \quad (1)$$

Equation 1 is strictly increasing and is for range 0 to ∞ but we need function that is with range from -∞ to +∞ as given under:

$$\frac{\tanh(x)+1}{2} = 1 - \frac{1}{e^{2x}+1} \quad (2)$$

$$\text{Bcoding} = \left(tanh\left(\sigma(a_n^7 W^H + b^H)\right) + 1\right)/2$$

$$= (tanh(a_n^H) + 1)/2 \quad (3)$$

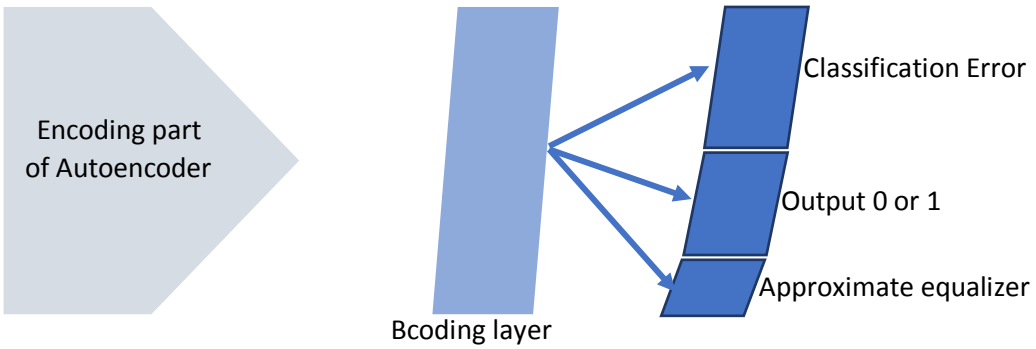

Fig. 3. Extracted features processed with binary code output

3.4 Loss Functions:

To learn hash functions needed useful information is provided by image labels that is also needed to classify images. Using this supervised information will make this method semi-supervised learning. To preserve semantic binary codes, extracted codes are trained with provided images label data that will not only map binary codes with image labels but also help to calculate accuracy. Using bcoding layer Q hidden attributes can be derived with each either 1 or 0. During training and testing classification is dependent on these hidden attributes as input image is associated with these binary valued outputs. Though we can calculate classification error using an optimization of a loss function. Using this procedure mapping of binary codes with semantically similar images can be ensured.

We assume a matrix WC $\epsilon\ \mathbb{R}^{Q\times M}$ to estimate mapping between image labels and binary hash codes. For any image I_n, y_n is original class label while y'_n is predicted class label. While calculating W_C, one of our objective function that needs to optimize is given as under:

$$\text{Loss1(W)} = \arg\min \sum_{m=1}^{M} L(y_n, \hat{y}_n) + \lambda \|W\|^2 \tag{4}$$

where L(.) is to minimize classification error that uses softmax outputs and minimizes cross-entropy error function. For multi-label classification, yn could have multiple entries as 1 that shows image association with multiple classes. For this purpose, we need to enlarge the margin of classification boundary so that multiple labels can be incorporated. To achieve this task network is set to give output like: for $y_{nm} = 1$ output will be $\hat{y}_{nm} \geq 1$ and for $y_{nm} = 0$ output will be $\hat{y}_{nm} \leq 0$ where y_{nm} is particular label for nth image and mth output node. Thus the loss function will become

$$L(y_{nm}, \hat{y}_{nm}) = \begin{cases} 0 & y_{nm} = 1 \wedge \hat{y}_{nm} \geq 1 \\ 0 & y_{nm} = 0 \wedge \hat{y}_{nm} \leq 0 \\ \frac{1}{2}|y_{nm} - \hat{y}_{nm}| & otherwise \end{cases} \tag{5}$$

Activations of each bcoding node should be approximate to $\{0, 1\}$. Though sigmoid function in last FC7 layer of convolutional autoencoder has given outputs between 0 and 1 but requires some constraint that can make these codes further approach to 0 or 1. Second part of loss function add constraint of maximizing the sum of squared errors with bcoding activations, that is, $\sum_{n=1}^{N} \|a_n^H - 0.5p\|^2$, where p is Q-dimensional vector having all 1s. This process ensures that our network will generate more appropriate binary code requirements.

Binary hashed codes should also have balance between 0s and 1s. To achieve this goal we need to minimize $\sum_{n=1}^{N} (mean(a_n^H) - 0.5)^2$, where mean(.) calculates the average of all the values in a vector. This constraint favors equal number of 0s and 1s in binary codes and expands the gap between 0s and 1s of binary hash codes.

Objective function by combining above mentioned equations to construct has codes with similarity preserving is given as under:

$$\arg min_w\ x \sum_{m=1}^{M} L(y_n, \hat{y}_n) + \lambda \|W\|^2 - y \sum_{n=1}^{N} \|a_n^H - 0.5p\|^2 + z \sum_{n=1}^{N} (mean(a_n^H) - 0.5)^2$$

where *x*, *y* and *z* is weightage given to each part of the objective function and can be selected at random.

4. Implementation:

This approach was implemented using open source library KERAS with Tensorflow as a backend. System used in this process equipped with ci7 processor 7th generation with 16gb ram and NVIDIA 1050 ti GPU. Software toolchain used to implement the model consist of IPython development environment using Keras 2.0 on Tensorflow backend and nvidia cuda 8.0. Training and testing data is manipulated in form of numpy arrays.

5. Experiments:

Experiments have been conducted on 4 most widely used datasets including Cifar-10, Mnist, Nus-Wide and Mirflicker. Furthermore, results have been compared with state of the art methods [1], [6]–[8], [11], [13, p.], [30], [38] that includes unsupervised and hashing techniques.

5.1 Datasets:

Above mentioned four datasets were divided into training, testing and validation sets. Description of these sets including dataset is given as follows:

CIFAR-10: This dataset has 10 categories having 32x32 color images. Each category has 6000 images, so in total we have 60000 color images in this dataset. 50000 images were selected at random for training while remaining were divided in testing and validation at random.

MNIST: 10 categories of grayscale images having hand written images from 0 to 9 with total 70000 images of 28x28 pixels. Training was performed using 60000 images selected at random but equal from each class. Remaining 10000 were used for testing and validation purposes.

NUS-Wide: This dataset is available with 81 semantic concepts with 21 most frequent and at least 5000 images per concept. Dataset contains 270000 images with each image associated with one or multiple labels. Around 90% data was randomly selected for training while remaining 10% was used for testing and validation purposes.

MIRFLICKR: Images from Flickr with multiple label having 38 semantic concepts and in total 25000 images. 24000 images were used for training while remaining were used for testing and validation selected at random.

Accuracy of the image retrieval was compared with other techniques on the basis of Mean Average Precision, Precision @top100, @top200, @top400, @top600, @top800 and j@top1000. Mean average precision (MAP) is the mean of average precision for every query. Average precision can be calculated using $Avg.Prec. = \frac{1}{M}\sum_{k=1}^{n}\frac{k}{M_k} \times rel_k$ where M is number of images relevant to query in database, M^k is number of relevant images in the top k returns and rel_k = 1 if the image ranked at kth position is relevant and 0 otherwise. Precisions at certain level of recall is known as Precision Recall Curves. It is calculated for all returned results. Average precision of top k returned images for each query is Precision @topk while average precision of the top 500 returned images for each query is precision @top500.

Table 1 (a). MAP scores with different lengths of hash codes

MAP	CIFAR10				MNIST			
	12bit	24bit	32bit	48bit	12bit	24bit	32bit	48bit
DHS	0.921	0.932	0.930	0.940	0.985	0.990	0.990	0.990
SSDH	0.801	0.813	0.812	0.814	0.975	0.982	0.982	0.982
DRSCH	0.721	0.733	0.726	0.747	0.951	0.953	0.955	0.966
NINH	0.600	0.696	0.689	0.702	0.931	0.949	0.958	0.959
CNNH	0.496	0.580	0.582	0.583	0.925	0.955	0.964	0.965
SDH-VGGF	0.363	0.528	0.529	0.542	0.542	0.938	0943	0.944
ITQ-VGGF	0.219	0.228	0.239	0.247	0.407	0.478	0.487	0.506
SH-VGGF	0.169	0.161	0.161	0.159	0.301	0.304	0.296	0.287
LSH-VGGF	0.132	0.124	0.144	0.157	0.176	0.191	0.220	0.305
SDH	0.255	0.330	0.344	0.360	0.526	0.915	0.921	0.926
ITQ	0.158	0.163	0.168	0.169	0.404	0.442	0.447	0.460
SH	0.124	0.125	0.125	0.126	0.290	0.278	0.260	0.254
LSH	0.116	0.121	0.124	0.131	0.162	0.236	0.222	0.276

Table 1 (b). MAP scores with different lengths of hash codes

MAP	NUS-WIDE				MIRFLICKR			
	12bit	24bit	32bit	48bit	12bit	24bit	32bit	48bit
DHS	0.801	0.802	0.802	0.810	0.850	0.845	0.860	0.890
SSDH	0.707	0.725	0.731	0.735	0.773	0.779	0.778	0.778
DRSCH	0.640	0.650	0.655	0.635	0.741	0.741	0.737	0.728
NINH	0.597	0.627	0.647	0.651	0.693	0.711	0.718	0.709
CNNH	0.536	0.522	0.533	0.531	0.667	0.688	0.654	0.626
SDH-VGGF	0.520	0.507	0.591	0.610	0.695	0.704	0.697	0.708
ITQ-VGGF	0.582	0.581	0.583	0.588	0.648	0.654	0.652	0.652
SH-VGGF	0.486	0.462	0.455	0.448	0.603	0.595	0.590	0.588
LSH-VGGF	0.432	0.451	0.464	0.466	0.571	0.574	0.580	0.589
SDH	0.414	0.465	0.451	0.454	0.595	0.601	0.608	0.605
ITQ	0.428	0.429	0.430	0.431	0.576	0.579	0.579	0.580
SH	0.390	0.391	0.389	0.390	0.561	0.562	0.563	0.562
LSH	0.404	0.384	0.394	0.400	0.557	0.564	0.562	0.569

5.2 Experiment Results and Analysis

MAP scores are shown in above **Table 1 (a)** and **(b)** that clearly elaborate the performance of our method in comparison to state of the techniques available in literature. For example, if we take 48-bit code for all datasets, SSDH is at 81%, DRSCH is at 74%, NINH is at 70% while our algorithm is producing 94% results. There are two main reasons, one is features that were extracted using convolutional autoencoder that ensures the reconstruction of original image from extracted features. Secondly semi-supervised loss function that improves search accuracy along with semantic similarity preservation. Same can be seen in the following graphs.

Training sets were used to train the model from scratch so that a fair comparison could be made between hashing and traditional techniques. Parameters extracted after training were given to bcoding layer that convert these feature vectors to binary hash code. With larger datasets having major portion as a training set leads to better feature extraction that ultimately results in good accuracy and better retrieval. Precisions with top retrieved images having 48-bit coding length is shown in **Table 2 (a)** and **(b)**. Same has been drawn in the form of graphs that elaborates success of our method in each dataset case.

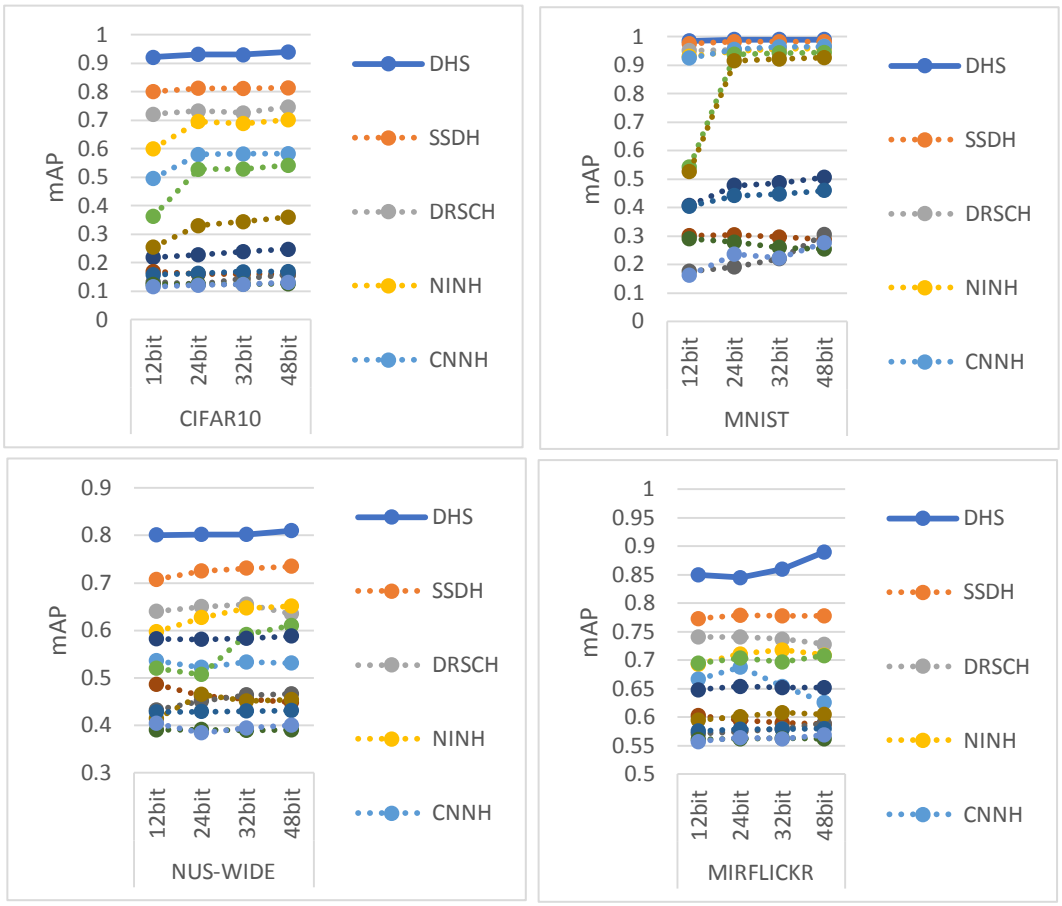

Fig. 4. mAP scores with different lengths of hashing code for four known datasets

MNIST is a simple and single labeled dataset that shows good performance in comparison to other datasets. Performance has been improved using our method as it has been shown in **Table II**. Classification performance for this single labeled dataset has been improved for all given code lengths. This shows that our technique retains good performance for lower dimensional features too.

CIFAR10 is a color image dataset having 10 different categories. Our technique has shown good results for this dataset too. Semantics have been retrieved in much details and accurately that we can reconstruct original like images from these extracted features. **Table II** has shown performance evaluation of this dataset with different number of retrieved images and proves effectiveness by producing better results.

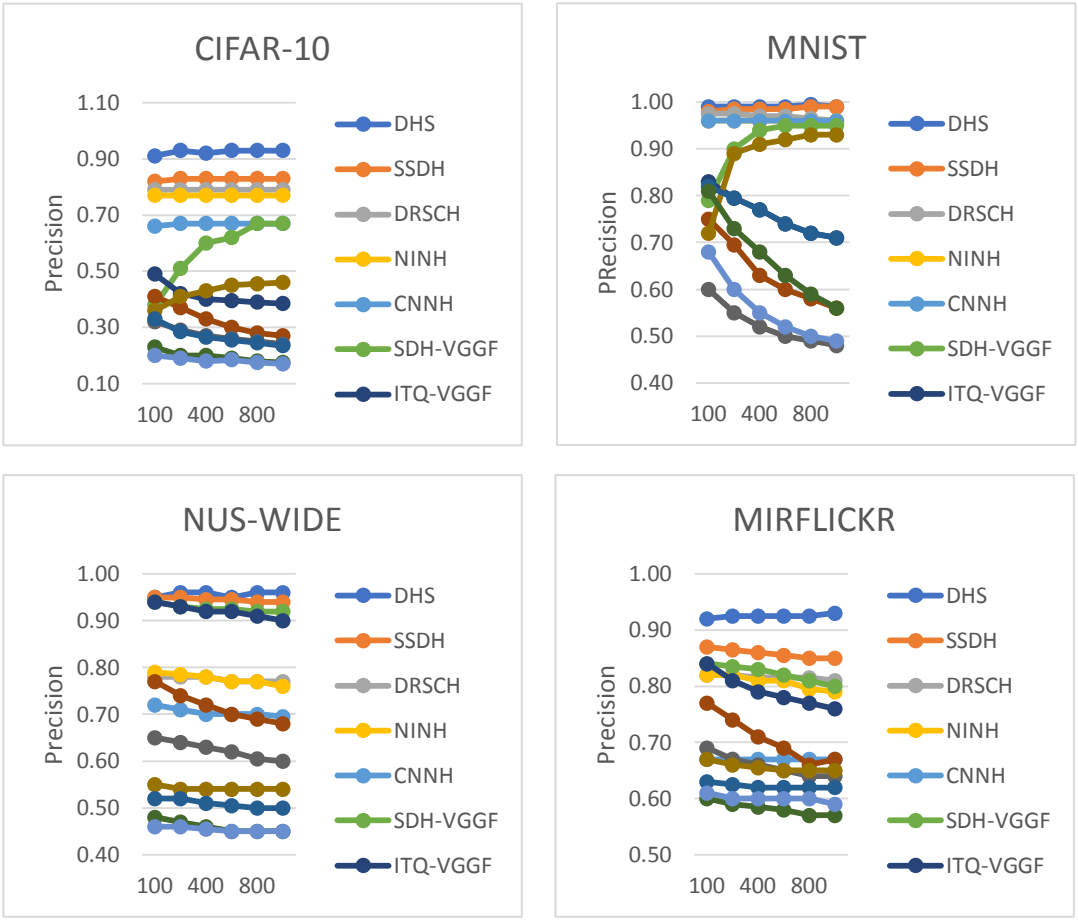

Fig. 5. Precision of retrieved images for top 100m 200, 400, 600, 800 and 1000 images

Table 2 (a). Precision scores at different outputs i.e. 100, 200, 400, 600, 800 and 1000

Precision	CIFAR10						MNIST					
	100	200	400	600	800	1000	100	200	400	600	800	1000
DHS	0.91	0.93	0.92	0.93	0.93	0.93	0.99	0.99	0.99	0.99	1.00	0.99
SSDH	0.82	0.83	0.83	0.83	0.83	0.83	0.98	0.99	0.99	0.99	0.99	0.99
DRSCH	0.79	0.79	0.79	0.79	0.79	0.79	0.98	0.98	0.97	0.97	0.97	0.96
NINH	0.77	0.77	0.77	0.77	0.77	0.77	0.96	0.96	0.96	0.96	0.96	0.96
CNNH	0.66	0.67	0.67	0.67	0.67	0.67	0.96	0.96	0.96	0.96	0.96	0.96
SDH-VGGF	0.38	0.51	0.60	0.62	0.67	0.67	0.79	0.90	0.94	0.95	0.95	0.95
ITQ-VGGF	0.49	0.42	0.40	0.40	0.39	0.39	0.83	0.80	0.77	0.74	0.72	0.71
SH-VGGF	0.41	0.37	0.33	0.30	0.28	0.27	0.75	0.70	0.63	0.60	0.58	0.56
LSH-VGGF	0.32	0.29	0.27	0.26	0.25	0.24	0.60	0.55	0.52	0.50	0.49	0.48
SDH	0.36	0.41	0.43	0.45	0.46	0.46	0.72	0.89	0.91	0.92	0.93	0.93
ITQ	0.33	0.29	0.27	0.26	0.25	0.24	0.82	0.80	0.77	0.74	0.72	0.71
SH	0.23	0.20	0.20	0.19	0.18	0.18	0.81	0.73	0.68	0.63	0.59	0.56
LSH	0.20	0.19	0.18	0.19	0.18	0.17	0.68	0.60	0.55	0.52	0.50	0.49

Table 2 (b). Precision scores at different outputs i.e. 100, 200, 400, 600, 800 and 1000

Precision	NUS-WIDE						MIRFLICKR					
	100	200	400	600	800	1000	100	200	400	600	800	1000
DHS	0.95	0.96	0.96	0.95	0.96	0.96	0.92	0.93	0.93	0.93	0.93	0.93
SSDH	0.95	0.95	0.95	0.95	0.94	0.94	0.87	0.87	0.86	0.86	0.85	0.85
DRSCH	0.78	0.78	0.78	0.77	0.77	0.77	0.82	0.82	0.82	0.82	0.82	0.81
NINH	0.79	0.79	0.78	0.77	0.77	0.76	0.82	0.82	0.81	0.81	0.80	0.79
CNNH	0.72	0.71	0.70	0.70	0.70	0.70	0.67	0.67	0.67	0.67	0.67	0.67
SDH-VGGF	0.94	0.93	0.93	0.93	0.92	0.92	0.84	0.84	0.83	0.82	0.81	0.80
ITQ-VGGF	0.94	0.93	0.92	0.92	0.91	0.90	0.84	0.81	0.79	0.78	0.77	0.76
SH-VGGF	0.77	0.74	0.72	0.70	0.69	0.68	0.77	0.74	0.71	0.69	0.66	0.67
LSH-VGGF	0.65	0.64	0.63	0.62	0.61	0.60	0.69	0.67	0.66	0.65	0.64	0.64
SDH	0.55	0.54	0.54	0.54	0.54	0.54	0.67	0.66	0.66	0.65	0.65	0.65
ITQ	0.52	0.52	0.51	0.51	0.50	0.50	0.63	0.63	0.62	0.62	0.62	0.62
SH	0.48	0.47	0.46	0.45	0.45	0.45	0.60	0.59	0.59	0.58	0.57	0.57
LSH	0.46	0.46	0.46	0.45	0.45	0.45	0.61	0.60	0.60	0.60	0.60	0.59

This method has also been tested for cross-domain images i.e. images that were not part of any dataset. As our aim was to have such a technique that can produce binary code having semantic information so that this code can be embed in image as annotation that will improve image retrieval procedure. This can be applied on big data as well as in security applications.

6. Conclusions:

A semi-supervised deep hashing technique has been presented in this paper that not only preserve semantic information but also produce binary hash codes based on extracted semantic features. This is achieved by using power of convolutional autoencoders which are able to recreate original like images from extracted semantic features. Extracted features after encoding procedure in CAE, features were presented to bcoding layer. This layer was trained using 3 different loss functions that ensures captivity of semantic features and calculate loss based on these loss functions. This technique is also scalable for many other datasets and databases. proposed technique was compared with state of the art techniques and it provides promising results. In future, this tchnique can be applied for live images with reduciton in processing time.

References

[1] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, “Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013. Article (CrossRef Link)

[2] G. Irie, Z. Li, X. M. Wu, and S. F. Chang, “Locally Linear Hashing for Extracting Non-linear Manifolds,” in *Proc. of 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2123–2130, 2014. Article (CrossRef Link)

[3] M. Kan, D. Xu, S. Shan, and X. Chen, “Semisupervised Hashing via Kernel Hyperplane Learning for Scalable Image Search,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 4, pp. 704–713, Apr. 2014. Article (CrossRef Link)

[4] J. He, W. Liu, and S.-F. Chang, "Scalable Similarity Search with Optimized Kernel Hashing," in *Proc. of Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, pp. 1129–1138, 2010. Article (CrossRef Link)

[5] B. Kulis and K. Grauman, "Kernelized Locality-Sensitive Hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, Jun. 2012. Article (CrossRef Link)

[6] Y. Weiss, A. Torralba, and R. Fergus, "Spectral Hashing," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., pp. 1753–1760, 2009. Article (CrossRef Link)

[7] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous Feature Learning and Hash Coding with Deep Neural Networks," *CoRR*, vol. abs/1504.03410, 2015. Article (CrossRef Link)

[8] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised Hashing for Image Retrieval via Image Representation Learning," in *Proc. of Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, Qubec City, Qubec, Canada, pp. 2156–2162, 2014. Article (CrossRef Link)

[9] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep Semantic Ranking Based Hashing for Multi-Label Image Retrieval," *CoRR*, vol. abs/1501.06272, 2015. Article (CrossRef Link)

[10] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep Hashing Network for Efficient Similarity Retrieval," in *Proc. of Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, pp. 2415–2421, 2016. Article (CrossRef Link)

[11] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," in *Proc. of Proceedings of the 25th International Conference on Very Large Data Bases*, San Francisco, CA, USA, pp. 518–529, 1999. Article (CrossRef Link)

[12] J. Wang and S. Chang, "Sequential Projection Learning for Hashing with Compact Codes," 2010. Article (CrossRef Link)

[13] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-Scalable Deep Hashing with Regularized Similarity Learning for Image Retrieval and Person Re-identification," *CoRR*, vol. abs/1508.04535, 2015. Article (CrossRef Link)

[14] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep Supervised Hashing for Fast Image Retrieval," in *Proc. of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2064–2072, 2016. Article (CrossRef Link)

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst. 25*, pp. 1106–1114, 2012. Article (CrossRef Link)

[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2014. Article (CrossRef Link)

[17] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015. Article (CrossRef Link)

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proc. of Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, pp. 580–587, 2014. Article (CrossRef Link)

[19] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," *CoRR*, vol. abs/1312.6229, 2013. Article (CrossRef Link)

[20] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *CoRR*, vol. abs/1605.06211, 2016. Article (CrossRef Link)

[21] A. Oliva and A. Torralba, "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope," *Int J Comput Vis.*, vol. 42, no. 3, pp. 145–175, May 2001. Article (CrossRef Link)

[22] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. of Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, Washington, DC, USA, pp. 886–893, 2005. Article (CrossRef Link)

[23] H. F. Yang, K. Lin, and C. S. Chen, "Supervised Learning of Semantics-Preserving Hash via Deep Convolutional Neural Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PP, no. 99, pp. 1–1, 2017. Article (CrossRef Link)

[24] Y. Zhang, K. Lee, and H. Lee, "Augmenting Supervised Neural Networks with Unsupervised Objectives for Large-scale Image Classification," *ArXiv E-Prints*, Jun. 2016. Article (CrossRef Link)

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. Article (CrossRef Link)

[26] J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun, "Stacked What-Where Auto-encoders," *ArXiv E-Prints*, Jun. 2015. Article (CrossRef Link)

[27] A. Makhzani and B. Frey, "Winner-Take-All Autoencoders," *ArXiv E-Prints*, Sep. 2014. Article (CrossRef Link)

[28] B. Kulis and T. Darrell, "Learning to Hash with Binary Reconstructive Embeddings," in *Proc. of Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, USA, pp. 1042–1050, 2009. Article (CrossRef Link)

[29] M. Norouzi and D. J. Fleet, "Minimal Loss Hashing for Compact Binary Codes," in *Proc. of Proceedings of the 28th International Conference on International Conference on Machine Learning*, USA, pp. 353–360, 2011. Article (CrossRef Link)

[30] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised Discrete Hashing," in *Proc. of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 37–45, 2015. Article (CrossRef Link)

[31] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2475–2483, 2015. Article (CrossRef Link)

[32] Y. Kang, S. Kim, and S. Choi, "Deep Learning to Hash with Multiple Representations," in *Proc. of 2012 IEEE 12th International Conference on Data Mining*, pp. 930–935, 2012. Article (CrossRef Link)

[33] Z. Xia, X. Feng, J. Peng, and A. Hadid, "Unsupervised Deep Hashing for Large-scale Visual Search," *ArXiv E-Prints*, Jan. 2016. Article (CrossRef Link)

[34] J. Wang, S. Kumar, and S. F. Chang, "Semi-Supervised Hashing for Large-Scale Search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012. Article (CrossRef Link)

[35] Q. Wang, L. Si, and D. Zhang, "Learning to Hash with Partial Tags: Exploring Correlation between Tags and Hashing Bits for Large Scale Image Retrieval," in *Proc. of Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, pp. 378–392, 2014. Article (CrossRef Link)

[36] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming Auto-Encoders," in *Proc. of Artificial Neural Networks and Machine Learning – ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*, T. Honkela, W. Duch, M. Girolami, and S. Kaski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 44–51, 2011. Article (CrossRef Link)

[37] M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun, "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition," in *Proc. of 2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007. Article (CrossRef Link)

[38] J. Zhang, Y. Peng, and J. Zhang, "SSDH: Semi-supervised Deep Hashing for Large Scale Image Retrieval," *CoRR*, vol. abs/1607.08477, 2016. Article (CrossRef Link)

Dr. Yasir Saleem is currently serving as an Associate Professor in University of Engineering and Technology (UET), Lahore, Pakistan. His research interests include Computer Vision, Image Processing, Computer networks, Information/Network Security, DSP, Power Electronics, Simulation and Control system. During his PhD, he did research work for one semester under supervision of Prof. Dr. Zainal Salam in Renewable Energy and Power Electronics Lab, Faculty of Electrical Engineering, UTM, Malaysia. He is an active research and currently supervising postgraduate students at MS and PhD levels. He has authored and co-authored journal and conference papers at national and international level in field of Electrical and Computer Science and Engineering.

Mr. Muhammad Khawar Bashir is working as Lecturer in Department of Statistics and Computer Science and holds MS degrees in Computer Science from National University (FAST), Lahore. Mr. Khawar's fundamental expertise is in the field of Image Processing and Computer Vision. Besides, other research areas of interest includes E-Commerce, Management Information Systems, Network Security and Machine Learning. Mr. Khawar has local and international publications. He presented research findings in international and national scientific conferences in various countries including China and Turkey. He has over ten years of teaching experiences. He is also member of different committees in UVAS.