# Deep Image Set Hashing

Jie Feng[1] Svebor Karaman[2] Shih-Fu Chang[1,2]
[1]Department of Computer Science, Columbia University
jiefeng@cs.columbia.edu
[2]Department of Electrical Engineering, Columbia University
svebor.karaman@columbia.edu, sfchang@ee.columbia.edu

## Abstract

*In applications involving matching of image sets, the information from multiple images must be effectively exploited to represent each set. State-of-the-art methods use probabilistic distribution or subspace to model a set and use specific distance measure to compare two sets. These methods are slow to compute and not compact to use in a large scale scenario. Learning-based hashing is often used in large scale image retrieval as they provide a compact representation of each sample and the Hamming distance can be used to efficiently compare two samples. However, most hashing methods encode each image separately and discard knowledge that multiple images in the same set represent the same object or person. We investigate the set hashing problem by combining both set representation and hashing in a single deep neural network. An image set is first passed to a CNN module to extract image features, then these features are aggregated using two types of set feature to capture both set specific and database-wide distribution information. The computed set feature is then fed into a multilayer perceptron to learn a compact binary embedding trained with triplet loss. We extensively evaluate our approach on multiple image datasets and show highly competitive performance compared to state-of-the-art methods.*

## 1. Introduction

With the ubiquity of camera network, ease of imaging and availability of online data, it is fairly easy to capture and access images in the form of a set. An image set is a collection of unordered images for a target, e.g. an object, a human face, an event etc. Images within a set could exhibit different characteristics about the target, such as different views of an object or a face, images of a scene taken under different lighting conditions, a set of video frames depicting different poses of a human action. Thus, an image set contains richer information than a single image and is potentially more useful for problems like object or scene classification, face recognition and action analysis. Many methods [38, 37, 24, 41, 15] have been proposed to leverage sets as input for matching problem. Most of these methods focus on set modeling and how to compute a proper matching distance between two sets. They usually don't have a compact representation for sets, making it expensive to store the set models in memory and slow to match, thus hard to scale when the number of targets increases.

Learning-based hashing [5, 23, 39, 22, 36] has received a lot of attention in problems like image retrieval. By computing a binary code for each image, much less memory space is needed and Hamming distance could be used to significantly reduce the matching time. Since the hash codes are learned, they are also able to preserve prior known constraints using supervised or semi-supervised learning and achieve good matching accuracy. Inspired by image based hashing, we consider the problem of hashing for image sets. While in the individual image case, each image is encoded as a binary code, we seek to represent a set as a single binary code regardless of the set size. This can effectively reduce the complexity of matching two sets without referring to the individual images composing them and can greatly reduce the time and memory cost during matching. Although the benefits of doing set hashing are obvious and appealing, it is not a trivial task: 1) a proper way of representing a set is needed to effectively integrate information from each image; 2) the hashing process should be connected with feature extraction so the image feature is optimized to achieve accurate hash code matching; 3) the method should work with sets of different sizes with the ability to improve performance when bigger sets are used.

In this paper, we propose a novel deep neural network model which takes an image set as input and computes a single compact binary code. Codes are optimized so that sets from the same class have smaller distances while sets from different classes have bigger distances. Each image is first passed through a feature extraction module to get an image feature. Then these features are aggregated to compute set features which summarize the features distribution within

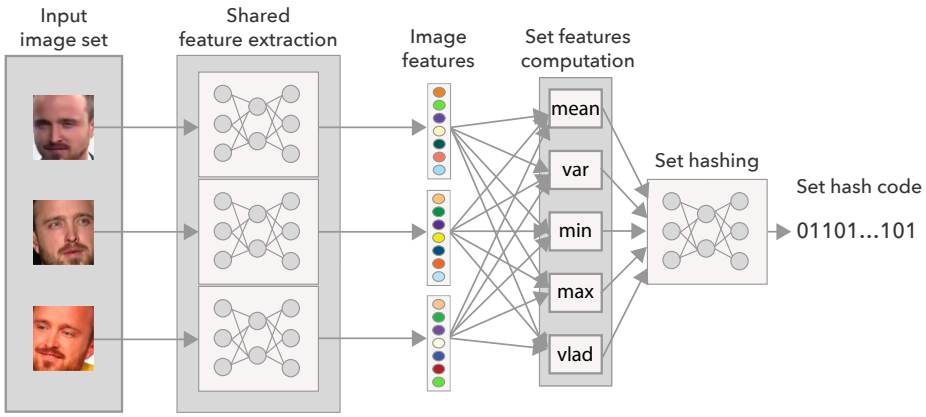


**Figure 1:** Our deep neural network architecture for set hashing. Each image in the input set goes through a shared feature extraction network producing one feature per image. All features are aggregated in the set feature layer before being encoded to produce a single binary hash code for the whole set.

the set. Finally, the set features are fed into a multilayer network to be transformed into a binary code. An overview of our set hashing network is given in Fig. 1. To ensure the learned codes are meaningful, the model is trained using triplet loss to get a metric binary embedding by directly optimizing on the distance value. This loss has been shown as an appropriate choice for metric learning [28, 25]. We evaluate the proposed method on multiple datasets used for image set matching, and show our method compares favorably with regards to the state-of-the-art including both single image hashing methods and set matching.

Our contributions can be summarized as follows:

1. We propose an end-to-end deep neural network structure which learns a binary code for a given image set. To the best of our knowledge, this is the first work to tackle set hashing problem using a deep learning approach.

2. We incorporated two types of complementary features to represent a set of image features - one capturing the local distribution geometry and the other encodes the set using a global dictionary. The combination of the two enables learning high quality set hashing codes.

3. Our extensive experiments show the proposed set hashing method achieves much better retrieval performance compared to state-of-the-art image hashing methods and is similarly effective compared to state-of-the-art set matching algorithms. As opposed to these state-of-the-art set matching method, ours is independent of the number of classes in the database thus making set based matching scalable.

## 2. Related works

We will first review here the related literature that studied how to represent and match image sets. Then we discuss works related to deep learning for hashing and set representation.

### 2.1. Image set representation and matching

The key problems for set matching are how to represent a set of images and how to properly compute the similarity between two given sets. The two being linked as the appropriate similarity measure depends on the set representation at hand. The existing works can be grouped into two categories: parametric and non-parametric methods.

Parametric approaches seek to represent the distribution within a set, for example using a single Gaussian [30] or Gaussian Mixture Models (GMM) [2] or as probability distribution functions (PDFs) using kernel density estimators [7]. The distribution representations are compared with f-divergence functions such as the Kullback-Liebler (KL) divergence. These approaches need to estimate a complex model with potentially many parameters for each set. Moreover they make strong assumption on the distribution within one set, and between gallery and query sets causing them to be sensitive to statistical noise.

The non-parametric methods seek a more effective set representation. Several works have focused on finding some representative exemplars: mean [38], affine hull or convex hull [4], approximated nearest neighbors [11], and regularized nearest points [42]. Other works use a geometric representation such as linear subspaces [6, 15] compared with the principal angles method [14] or the projection kernel metric [6]. Subspace based methods perform well when each set represents a dense sampling, but tends to struggle when the set is of small size with complex data variations. Lin-

ear subspace models are usually estimated from an eigen-decomposition of the covariance matrix but discarding non leading eigenvectors. Hence, the covariance matrix characterizes the set structure more faithfully and has therefore also been used as a set representation [37] but its higher dimensionality is a burden when dealing with large scale problems. In the Hashing across Euclidean space and Riemannian manifold (HER) framework proposed in [21], video clips are encoded as covariance matrices and embedded into a Reproducing Kernel Hilbert Space (RKHS) before applying a SVM-based hash learning procedure.

### 2.2. Deep learning for single image hashing

In the last years, several approaches were proposed for binary embedding with deep networks. A two stage approach was developed in [39] where the authors first learn target codes using pair similarity supervision and then train a deep hash network to map each sample to its target code. The pre-training procedure renders the method not scalable. End-to-end deep hash methods were proposed in [22, 17, 44]. The authors of [22] proposed to add a latent hash layer to a standard AlexNet and use a classification loss to train their model. As our goal is to learn hash codes for retrieval of similar samples, the optimization should focus on learning hash codes for which the hamming distance preserves similarity ranking. The classification optimization of the previous work has no guarantee to do so, hence recent methods prefer to rely on a triplet loss to enforce hamming distances between hash codes to respect the classes semantics [44]. The authors of [17] also proposed a divide and encode approach to improve bit independence. All these methods proposed solely deep hashing methods that produce a hash code for each single image separately.

### 2.3. Deep learning for set classification

A few works have addressed the problem of set classification with deep networks, mainly using class specific models. The authors of [24] propose a multi-manifold deep metric learning approach for image set classification. Class-specific neural networks are trained using a maximal manifold margin that minimizes the intra-manifold and maximizes the inter-manifold distances. In [8], class specific auto-encoders are trained and the classification procedure seeks the minimal reconstruction error when auto-encoding each sample of the test set with each class model, the set label being estimated as the most recurring label amongst all images of the test set. A similar classification strategy is used in [29] where class-specific neural networks are trained using a Pooled Convolutional representation as input. In these methods, the classification of one test set requires passing each test image through each class network. Hence, the complexity of these methods grows linearly with the number of classes making them intractable for problems with large number of classes in the gallery. Furthermore, the test set structure is only used in aggregation of sample distances or classification results.

To the best of our knowledge, our method is the first to tackle the set hashing problem in an end-to-end learning framework. We specifically aim for a single model that can efficiently encode sets of images from a number of classes without relying on class-specific models that will prevent efficient scalability when the number of target classes grows. Our model enables large-scale set based retrieval thanks to the binary representation.

## 3. Deep Set Hashing Network

We design a deep neural network model to process input in the form of image sets. The network structure is composed of three blocks: an image feature extraction phase that is applied to each sample, an aggregation layer applied to all the image features within an input set to compute set features, and finally a hashing phase that encodes the set features into a single hash code. The network structure is illustrated in Figure 1. We detail in this section each building block of our network.

### 3.1. Image feature extraction

Each image $I_i$ in the input set $\mathcal{S}$ first goes through a feature extraction module to transform the original pixels to a powerful feature representation. Convolutional Neural Networks (CNN) are used as the feature extractor given its proven success on various computer vision problems. As has been showed in various published works, the CNN structure needs to be engineered to work on specific tasks and data domains.

To make the image feature relevant and useful, we could choose a network design that has been shown to work well for a particular problem. VGG-16 [31], Inception network [33] and the recent ImageNet-winner Residual network [10] achieve excellent performance on object classification problem. They would be the preferred architectures for feature extraction when we work with object related datasets. For face recognition, CASIA network [43], Google FaceNet [28] and Oxford VGG-face [25] are some successful examples. Once the feature extraction part is chosen, it is shared among all images. In Section 3.4, we will also discuss how the network parameters (e.g., coefficients) will be learned in an end-to-end manner to derive the optimal set hashing method.

### 3.2. Set feature computation

Given a set of input samples, we need to find a way to merge them into a unified set representation regardless of how many samples are present for a given set. This process enables us to model a set compactly and perform distance

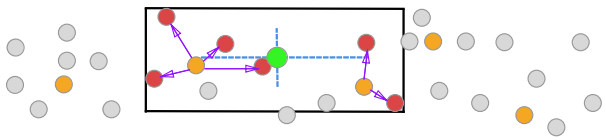

**Figure 2:** Illustration of set features in the image feature space: red nodes are images of the set $\mathcal{S}$ considered, gray node are images not in the set, orange nodes are cluster centers of the VLAD dictionary, the green node is $mean(\mathcal{S})$, black lines represent the space delimited by $min(\mathcal{S})$ and $max(\mathcal{S})$, blue lines represent $var(\mathcal{S})$ and purple arrows represent $VLAD(\mathcal{S})$ (best viewed in color). Please see text for more details.

measure without resorting to the original images. Meanwhile, information from different images should be captured in the final representation so we benefit from having more data than a single image.

To ensure the set feature is generic to maintain characteristics of the unknown underlying image feature distribution, we derive statistics to describe the global geometry property of the set distribution. Specifically, for each input set $\mathcal{S}$ we compute:

- the average feature: $mean(\mathcal{S}) \in \mathbb{R}^d$;
- the variance of each feature dimension: $var(\mathcal{S}) \in \mathbb{R}^d$;
- the element-wise minimum feature: $min(\mathcal{S}) \in \mathbb{R}^d$;
- the element-wise maximum feature: $max(\mathcal{S}) \in \mathbb{R}^d$.

$mean(\mathcal{S})$ indicates the average location of the set in the sample feature space. $var(\mathcal{S})$ shows how values of each feature dimension changes relative to its mean within the given set. $min(\mathcal{S})$ and $max(\mathcal{S})$ give the range of the set features. The last three measures capture the shape of the set in the feature space. An illustration of the set statistic feature is shown in Fig 2. Here each gray node represents an image in the feature space, red nodes denote a set of images. Green node is the mean of the set, the lines surrounding the set gives $min(S)$ and $max(S)$. $var(S)$ describes the average variation of the set around the mean. From the graph, we can see these summary statistics are useful to provide an overall view of the set, but they only depict the set itself independent of other sets.

For an effective comparison, it is necessary to derive features based on the overall feature distribution across all set samples. Inspired by the popular pooling methods of image local descriptors in image retrieval such as Bag-of-words (BoW) [32] and its improved derivations, we add another set feature using the Vector of Locally Aggregated Descriptors (VLAD) [12] representation which has been shown very effective in describing a set of features with a lower dimension compared to Fisher Vectors (FV) [26]. In our case, we use the training image pool to construct the dictionary $D$ that is required in computing VLAD features. This dictionary no longer encodes local image patches but the higher level concepts at the image level. Given the deep features extracted from the CNNs, the dictionary can be used to represent semantic structure like pose, attributes, category etc. To compute the dictionary, we first extract CNN features for a random subset of the training images, then perform K-Means to get the cluster centroids: $\mathcal{C} = \{c_1, \ldots, c_k\}$. Assuming a set $S$ of $N$ images $\{I_1, \ldots, I_N\}$ with corresponding CNN features $\{x_1, \ldots, x_N\}$, each component $v_{k,j}$ of the VLAD descriptor for $\mathcal{S}$ is calculated as:

$$v_{k,j} = \sum_{i=1}^{N} w_k(x_i)(x_{i,j} - c_{k,j}), \tag{1}$$

where $k \in [1, K]$ is the cluster index in $D$ and $j \in [1, d]$ indexes the dimension of the CNN image feature $x$. $w_k(x_i)$ is the weight for feature $x_i$, which is 1 if $NN(x_i) = c_k$, otherwise it's 0. L2-normalization is applied to get the final VLAD feature $VLAD(S)$.

However, naively compute the VLAD feature has several limitations. First, the hard assignment of centroid is not differentiable, making it hard to train end-to-end. Second, the dictionary is static once computed, making it not compatible with the iterative training process of a deep network. To handle these issues, we make two key modifications for VLAD extraction. Instead of assigning each sample to one centroid, soft assignment is used based on the distance to each centroid. This is computed as:

$$w_k(x_i) = \frac{e^{-||x_i - c_k||^2}}{\sum_k e^{-||x_i - c_k||^2}}, \tag{2}$$

which essentially is a softmax operation. The new weight now is a real value between 0 and 1 with higher value assigned to closer centroid. Soft assignment also eliminates the problem of sparseness when a small number of images are presented in a set. The second modification comes for the training process. With the change of network parameters, the deep feature for each image is also changed. It is necessary that the dictionary is recomputed to incorporate the updated features. Since the clustering is relatively time consuming, it is not practical to do it for each batch. To strike a balance, we compute a new dictionary at the beginning of each epoch and use it to encode VLAD. This keeps the training efficient while making VLAD adaptive to the updated features. We choose a dictionary size of 64 empirically for a good trade-off between accuracy and computational complexity.

As shown in Fig. 2, orange nodes denote cluster centers and purple arrows denote the difference vector between each set node and its closest cluster center, we can see

VLAD describes the distribution pattern of set samples relative to the whole image feature space, thus giving additional information on the set.

The output of the aggregation layer for a set $\mathcal{S}$ is then the concatenation of the five set features $F(\mathcal{S}) = (mean(\mathcal{S}), var(\mathcal{S}), min(\mathcal{S}), max(\mathcal{S}), VLAD(\mathcal{S}))$. Notice unlike other existing methods where set feature and image feature are usually computed independently, in our case, by connecting image feature extraction with set feature computation, the image feature will be learned through the end-to-end back propagation training process described in Section 3.4 below, to optimize the set feature, thus no fixed assumption of the set distribution is made.

### 3.3. Set Hashing

Hashing is performed by multiple fully connected layers with non-linear activation. We use $W^i$ to indicate the parameter of $i$th layer and $b^i$ is the corresponding bias. Given a set aggregate feature $F(\mathcal{S})$, the output of the first layer could be written as: $h_1(F(\mathcal{S})) = s(W^1 F(\mathcal{S}) + b^1)$ where $s(.)$ is a nonlinear activation function. For all layers except the last layer, rectified linear unit (Relu) is used while the sigmoid activation is used in the last layer to get output value between 0 and 1. Similarly, output of the $i$th layer is: $h_i(F(\mathcal{S})) = s(W^i h_{i-1}(F(\mathcal{S})) + b^i)$. Note that each layer $h_i$ is a multi-dimensional vector. Assuming we use $M$ layers for hashing, our set hashing network output is:

$$H(F(\mathcal{S})) = h_M(F(\mathcal{S})) = s(W^M h_{M-1}(F(\mathcal{S})) + b^M) \tag{3}$$

To generate the final binary code, we use $0.5$ to threshold this output: $B(\mathcal{S}) = sgn(H(F(\mathcal{S})) > 0.5)$.

### 3.4. Training the network

One of the distinct novelty of our proposed approach is the end-to-end architecture that allows us to learn different parts (image feature, set aggregation, and hashing) in the same model. To train our network, we need to define a proper loss function given set training data and labels. In our case, we want the final binary code to preserve set relations defined by corresponding labels, i.e. codes of sets with the same labels should have a smaller Hamming distance while codes of sets with different labels should have a bigger Hamming distance. To train such objective, the triplet loss could be used as in [28] and [25].

Let's denote a set triplet $t_i$ as $< \mathcal{S}_a, \mathcal{S}_p, \mathcal{S}_n >$ with corresponding feature for each set as $F(\mathcal{S}_a)$, $F(\mathcal{S}_p)$ and $F(\mathcal{S}_n)$. Here $\mathcal{S}_a$ is the anchor set, $\mathcal{S}_p$ is the positive set which has the same class label as $\mathcal{S}_a$, $\mathcal{S}_n$ is the negative set with different class label as $\mathcal{S}_a$. To make the notation simpler, we replace $F(\mathcal{S}_a)$ by $F_a$, $F(\mathcal{S}_p)$ by $F_p$ and $F(\mathcal{S}_n)$ by $F_n$. We get the output from the final hash layer and compute the loss

(a) Input image (b) Cropping (c) Scaling (d) Flip

**Figure 3:** Examples of augmented images.

for each triplet:

$$L_{t_i} = max\{0, ||H(F_a) - H(F_p)||_2^2 - ||H(F_a) - H(F_n)||_2^2 + \alpha\} \tag{4}$$

where $||.||_2^2$ is the squared Euclidean distance. The overall triplet loss then could be computed as the average of all individual triplet losses:

$$J_0 = \frac{1}{|\mathcal{T}|} \sum_{t_i \in \mathcal{T}} L_{t_i} \tag{5}$$

Here $\mathcal{T}$ is the collection of all training triplets. Since the hard binary output are non-differentiable, we use the real valued outputs from the sigmoid activation. To help learn better codes, we introduce two additional cost. First, we compute a distance between real valued output from last hash layer and binary output:

$$J_1 = \frac{1}{2}||\mathbf{B} - \mathbf{H}||_F^2 \tag{6}$$

where $\mathbf{B}$ is the matrix of binary outputs for all input samples, $\mathbf{H}$ is the real valued outputs matrix and $||.||_F^2$ is the squared Frobenius norm. This binary approximation cost helps push the real valued output to be close to binary output. Second, we would like the learned codes to be balanced, meaning the variance of the output values should be maximized:

$$J_2 = \frac{1}{N} tr(\mathbf{H}\mathbf{H}^T) \tag{7}$$

Here $N$ is the total number of samples, i.e. $N = 3|\mathcal{T}|$. Overall, our loss function is defined as:

$$J = J_0 + \lambda_1 J_1 - \lambda_2 J_2$$

$\lambda_1$ and $\lambda_2$ are parameters to balance the different cost terms, they were set empirically to 1 and 0.1 for all experiments based on preliminary evaluation. The optimization is carried out with Stochastic Gradient Descent (SGD). Since our architecture is actually a single network, the back-propagation will optimize the parameters in both the set hashing layers and the image feature extraction layers. The set features are parameter-free and allow the gradient to be back-propagated to the image feature extraction block.

### 3.5. Prepare Sets for the Network

Set inputs to our network could have very different sizes. When the set size is small, our set feature, especially statistics like mean and variance will not be meaningful. In this

| Methods | **MNIST** (32bits) | **CIFAR10** (32bits) |
|---|---|---|
| ITQ [5] | 0.82 | 0.27 |
| ITQ (deep feature) | 0.97 | 0.64 |
| KSH [23] | 0.98 | 0.37 |
| KSH (deep feature) | 0.97 | 0.65 |
| Deep image hashing [40] | 0.97 | 0.52 |
| Deep image hashing [22] | 0.98 | 0.64 |
| Set hashing | **0.99** | **0.83** |
| Unbinarized set feature | 0.99 | 0.85 |

**Table 1:** Mean Average Precision (MAP) of different hashing methods on MNIST and CIFAR10.

case, it is necessary to augment the set to include more images. We employ 3 types of augmentation to introduce variations: 1) cropping: use a fixed size window to generate random cropped images; 2) scaling: downsample the image to a lower resolution then resize it to the original size; 3) flip: transform the image with a vertical or horizontal symmetry. Example images are shown in Fig 3. These types can be combined to generate many extra images so we can have a reasonably big set. This augmentation process brings more variation to the training process, making the learned model more robust to changes and noise present in the data. Besides the issue of small sets, it is possible that we do not have a sufficient number of sets for training, in this case, random subsets are sampled from the original set to expand the training data. The generated subset shares the same set label.

# 4. Experiments

To evaluate our proposed set hashing method, we perform various types of experiments. First, we compare retrieval performance between our set hashing and different traditional individual image hashing techniques to show that our method can effectively use set input to achieve much better performance. Second, we compare with state-of-the-art set matching algorithms and show our learned set codes can achieve similar performance with a much smaller memory footprint for the representation and matching complexity. Third, we evaluate the influence of some important parameters of our network.

## 4.1. Comparison with individual image hashing

Two standard hashing datasets are used: MNIST and CIFAR10. MNIST [18] consists of 70000 handwritten digit images from 10 classes (0-9). Each image is grayscale of size 28x28 pixels. 50000 images are used as training data and the remaining 20000 images are used as testing data. CIFAR10 [16] is a subset of the Tiny image dataset [34]. It contains 10 object classes, with a total 60000 images. Each image is 32x32 pixels in color. We use 50000 images for training and 10000 for testing. Since these two datasets have no set information, we construct our own sets. For each image in the training data, we randomly sample additional images of the same class to form a set of size 10. This ensures each image appears at least once in training sets. This is also used to form query sets of size 30.

For image based hashing, the performance is measured by computing set distance as the mean average image distance. We compare hash codes using the Hamming distance. During evaluation, testing data is used as query and training data is used as gallery to perform retrieval. Mean Average Precision (MAP) is used as the performance evaluation metric. Each image set in the test set is used as a query and when the retrieved image set belong to the same class, the result is considered correct.

If not specified otherwise, raw pixels are taken as input to learn the codes. The deep features used by ITQ and KSH are extracted from CNNs trained on each dataset for classification using the same structure as the feature extraction part of our model. Details about the CNN architectures used for each dataset are given in Section 4.4. We use two layers for our set hashing, with 512 and 32 nodes respectively, which hence produce 32 bits codes.

Comparisons are done with both standard hashing methods and deep learning based hashing methods, results are shown in Table 1. Given that MNIST is a simple dataset which contains limited variation for each character, all methods perform quite well with deep learning based approaches reaching very high MAP value close to 1. Our set hashing is even able to improve slightly to 0.99. On CIFAR10, all methods perform lower than on MNIST since CIFAR10 is more challenging with low quality images of objects under large appearance variation. KSH and ITQ trained directly on pixels perform poorly. By using deep features, their performance is greatly improved. However, an end-to-end feature learning and hashing network for image [22] achieves a higher MAP of 0.64. Our set based hashing method further increases the performance beating image based hashing methods by a large margin, indicating the effectiveness of our method in encoding multiple images within a set into a single binary code. Additionally, we also evaluate performance of the learned hash codes before binarization in the last row of Table 1. The real valued feature are compared using the Euclidean distance. They achieves better performance than hash codes but not with a large margin, indicating the binarization has a minimal effect on the performance thanks to the binary approximation cost $J_1$ defined in Eq. 6.

## 4.2. Comparison with set matching

We use some of the most popular image set data to compare with non-hashing based set matching methods. Two face datasets are used, including a small scale Honda/UCSD dataset [19] and the larger scale YouTube Celebrity [13].

| Methods | **Honda/UCSD** | **YTC** | **ETH-80** |
|---|---|---|---|
| DCC [15] | 92.56 | 51.42 | 91.75 |
| MMD [38] | 92.05 | 54.04 | 77.50 |
| AHISD [4] | 91.28 | 61.49 | 78.75 |
| CHISD [4] | 93.62 | 60.42 | 79.53 |
| SANP [11] | 95.13 | 65.60 | 77.75 |
| CDL [37] | 98.97 | 56.38 | 77.75 |
| RNP [42] | 95.90 | 65.82 | 96.23 |
| ADNT [8] | 100 | 71.35 | 98.12 |
| RT [9] | 100 | 74.10 | 95.50 |
| IDLM [29] | 100 | **76.52** | **98.64** |
| Set Hashing (512 bits) | **100** | 75.03 | 97.23 |

**Table 2:** Average classification accuracy on image set datasets, compared with state-of-the-art image matching techniques.

For object set matching, the ETH-80 dataset [20] is used.

The Honda/UCSD dataset contains a total of 59 video sequences from 20 different subjects. Each video has a frame count ranging from 12 to 645. The original face images have a resolution of 32x32 pixels which are very low quality, they are resized to 100x100 to be fed into our face feature extraction neural network. YouTube Celebrities (YTC) dataset consists of 1910 videos of 47 celebrities. The face images exhibit large variations in illumination, pose and expression. The face region of each frame is extracted using an off-the-shelf face detector [35]. For each person, three videos are randomly selected as the training data (which is used as the gallery during testing), the remaining videos are used as test queries. The ETH-80 object dataset contains image sets of 8 object categories: apples, cars, cows, cups, dogs, horses, pears and tomatoes. Each category has ten object instances and each instance has images under 41 different orientations. We use 5 instances as training data (which become gallery sets during testing), and the remaining 5 are used as test query sets.

For evaluation, we use classification accuracy as metric to be comparable with both set matching and classification approaches. The nearest neighbor is used to predict the label. In this experiment, we use two hashing layers with 1024 and 512 nodes. This configuration works best across all datasets. We summarize the results in Table 2. All recently proposed methods [8, 9, 29] perform well on these datasets. On Honda/UCSD dataset, four methods are able to achieve perfect classification including ours. Our method performs slightly worse on ETH-80 than ADNT and IDLM but still much better than all other methods with only 512 bits for each set. YTC is more challenging given face images captured in vastly difference conditions. Our method is able to perform second best with a little lower accuracy than IDLM. We would like to point out that, as discussed in Section 2.3, both IDLM and ADNT train class specific deep models. Therefore, the complexity in training and testing linearly increases according to the number of classes. At test time, each image in the query set has to be processed independently by all classes models and voting is needed to decide the final set label. Our method uses a single model for all classes and produces one hash code regardless of how many images are in each set. Thus, the complexity of our approach does not grow as the number of classes increases and testing is much faster. Furthermore, our model learns a similarity-preserving binary embedding and thus could also handle matching for classes out of the training set.

| Methods | 16 bits | 32 bits | 64 bits |
|---|---|---|---|
| ITQ (image) | 0.24 | 0.27 | 0.28 |
| ITQ (deep feature) | 0.62 | 0.64 | 0.66 |
| KSH (image) | 0.31 | 0.37 | 0.39 |
| KSH (deep feature) | 0.63 | 0.65 | 0.66 |
| Deep Hashing [22] | 0.61 | 0.64 | 0.67 |
| Set Hashing | **0.76** | **0.83** | **0.87** |

**Table 3:** Mean Average Precision (MAP) of different hash bits on CIFAR10 dataset.

## 4.3. Effects of key configurations

There are several important configurations in our network which will influence the final output. Here we conduct a detailed analysis on each factor. To do that, we use CIFAR10 as a testbed given it is easy to form sets and has enough data to train our model from scratch. Settings similar to Section 4.1 are applied with modifications for each experiment conducted. The results are reported by averaging 10 trials of random set generation. For set hashing, we are interested in the following questions: 1) the performance under different number of hash bits; 2) how different set sizes will affect our method; 3) how effective is the augmentation when applied to small sets; 4) how useful each type of set feature is.

### 4.3.1 Hash Bits Comparison

We evaluate the impact of hash bits number for the matching performance. The results are given in Table 3. From the table, it can be seen that the performance of each method grows consistently with the number of bits. Due to the use of deep features, KSH and ITQ performs similarly to the end-to-end image based deep hashing which essentially only uses a simple linear transformation and sigmoid activation for the compression. Our set hashing however offers much better performance across all hash bits selection.

### 4.3.2 Influence of set size

**Set size**: to evaluate the effect of different set sizes, we construct query sets with sizes in the range of values $V$ = [1, 5,

| Input set size | No augmentation | Small augmentation | Large augmentation |
|---|---|---|---|
| 1 | 0.25 | 0.41 (5) | 0.62 (10) |
| 5 | 0.55 | 0.70 (10) | 0.69 (20) |
| 10 | 0.69 | 0.73 (20) | 0.74 (30) |
| 20 | 0.75 | 0.80 (30) | 0.77 (40) |

**Table 4:** Mean Average Precision (MAP) of different set sizes on CIFAR10 dataset. The augmented set size is shown in parenthesis.

10, 20] while keeping the gallery sets size fixed to 10 samples and using hash codes of 32 bits. The results can be seen in the second column of Table 4. The performance increases consistently when the query set size increases, which indicates our method is able to leverage more data to improve its performance.

**Data augmentation**: we also investigate how data augmentation, as described in Section 3.5, can improve our set hashing framework performance especially when dealing with small set sizes. Two augmentation cases are defined. Small augmentation expands $V$ to $S(V)$=[5, 10, 20, 30] and large augmentation expands $V$ to $L(V)$=[10, 20, 30, 40]. The results reported in Table 4 show that data augmentation can significantly improve the set hashing performance, especially with small query sets (MAP from 0.25 to 0.62 with a single image input set). For larger set sizes, augmentation adds small improvement indicating the performance tends to converge when sufficient number of samples are used.

#### 4.3.3 Set feature comparison

We evaluate the performance using each type of set features (statistics and VLAD) and their combination. By only using one of the set features, our method is able to get reasonably good results with statistics performing better than VLAD feature in this particular case with MAP 0.80 and 0.79 respectively. However, combining these two features the MAP gets to 0.83. The result shows the complementary property of these two set features, they capture the local characteristics and the global context of a set, respectively.

### 4.4. Implementation details

Based on the target datasets, we use different pre-trained CNN model as feature extraction module. For experiments on face dataset, we use the CASIA face model [43]. The output of pool5 layer is treated as feature. We replaced Relu with parametric Relu as activation function which gives a little improvement in performance. For the object dataset, we use the VGG-16 [31] model which won ILSVRC-2014 competition [27]. For MNIST and CIFAR10, we use custom designed network structures. MNIST network uses two $5 \times 5$ convolution layers, each followed by a $2 \times 2$ max pooling layer. A fully connected layer with 256 nodes is used to compute the feature. For CIFAR network, 3 convolution with max pooling layers are used and followed by a fully connected layer for feature extraction. For all the datasets except MNIST and CIFAR10, we expand each original image set by generating 50 subsets. If the set is small, we perform augmentation using the random transformations described in Section 3.5 to make the set at least 10 images, then do the sampling. Training is performed with 3000 triplets in each epoch and run for 10000 epochs. The triplet margin $\alpha$ is set to $\frac{1}{2}\sqrt{b}$ where $b$ is the number of bits used in a given experiment, this setting makes the margin grow sub-linearly with increasing binary code sizes. We implemented our approach with Theano [3] and TensorFlow [1] and the training is performed on a NVIDIA GTX Titan X.

## 5. Discussion

In this work, we tackle the challenging problem of hashing image sets for scalable matching. We introduce a deep network which takes an image set as input, and process it with image feature extraction, computing set feature and finally convert it to a single binary code. The network is learned from end-to-end where triplet loss is used to optimize the hashing output and feature learning. Extensive experiments are conducted showing our set based hashing is superior to single image based hashing which is also trained using deep models. Our method is also able to achieve similar retrieval performance with state-of-the-art set matching algorithms which does not have a compact representation as binary codes and use more complicated class-specific models. Further experiments show our approach is able to improve with larger set size and the set features we used are complementary and work better when used together.

Set hashing is a valid solution to a real-world problem when dealing with large collection of images. Our future effort will be dedicated to the design of a network architecture which can potentially learn set features and domain specific set data generation.

## 6. Acknowledgements

This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2014-14071600012. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

## References


[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] O. Arandjelović, G. Shakhnarovich, J. Fisher, R. Cipolla, and T. Darrell. Face recognition with image sets using manifold density divergence. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 581–588. IEEE, 2005.

[3] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[4] H. Cevikalp and B. Triggs. Face recognition based on image sets. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2567–2573. IEEE, 2010.

[5] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(12):2916–2929, 2013.

[6] J. Hamm and D. D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the 25th international conference on Machine learning*, pages 376–383. ACM, 2008.

[7] M. Harandi, M. Salzmann, and M. Baktashmotlagh. Beyond gauss: Image-set matching on the riemannian manifold of pdfs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4112–4120, 2015.

[8] M. Hayat, M. Bennamoun, and S. An. Learning non-linear reconstruction models for image set classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1914, 2014.

[9] M. Hayat, M. Bennamoun, and S. An. Reverse training: An efficient approach for image set classification. In *Computer Vision–ECCV 2014*, pages 784–799. Springer, 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[11] Y. Hu, A. S. Mian, and R. Owens. Sparse approximated nearest points for image set classification. In *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*, pages 121–128. IEEE, 2011.

[12] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010.

[13] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley. Face tracking and recognition with visual constraints in real-world videos. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[14] T.-K. Kim, O. Arandjelović, and R. Cipolla. Boosted manifold principal angles for image set-based recognition. *Pattern Recognition*, 40(9):2475–2484, 2007.

[15] T.-K. Kim, J. Kittler, and R. Cipolla. Discriminative learning and recognition of image set classes using canonical correlations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1005–1018, 2007.

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009.

[17] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3278, 2015.

[18] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.

[19] K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman. Video-based face recognition using probabilistic appearance manifolds. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–313. IEEE, 2003.

[20] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–409. IEEE, 2003.

[21] Y. Li, R. Wang, Z. Huang, S. Shan, and X. Chen. Face video retrieval with image query via hashing across euclidean space and riemannian manifold. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4758–4767, 2015.

[22] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 27–35, 2015.

[23] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2074–2081. IEEE, 2012.

[24] J. Lu, G. Wang, W. Deng, P. Moulin, and J. Zhou. Multi-manifold deep metric learning for image set classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1137–1145, 2015.

[25] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. *Proceedings of the British Machine Vision*, 1(3):6, 2015.

[26] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[28] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[29] S. A. A. Shah, M. Bennamoun, and F. Boussaid. Iterative deep learning for image set based face and object recognition. *Neurocomputing*, 174:866–874, 2016.

[30] G. Shakhnarovich, J. W. Fisher, and T. Darrell. Face recognition from long-term observations. In *ECCV 2002*, pages 851–865. Springer, 2002.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[34] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):1958–1970, 2008.

[35] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[36] J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to hash for indexing big data - a survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.

[37] R. Wang, H. Guo, L. S. Davis, and Q. Dai. Covariance discriminative learning: A natural and efficient approach to image set classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2496–2503. IEEE, 2012.

[38] R. Wang, S. Shan, X. Chen, and W. Gao. Manifold-manifold distance with application to face recognition based on image set. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[39] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, volume 1, page 2, 2014.

[40] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, volume 1, page 2, 2014.

[41] O. Yamaguchi, K. Fukui, and K.-i. Maeda. Face recognition using temporal image sequence. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 318–323. IEEE, 1998.

[42] M. Yang, P. Zhu, L. Van Gool, and L. Zhang. Face recognition based on regularized nearest points between image sets. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, pages 1–7. IEEE, 2013.

[43] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[44] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1556–1564, 2015.