Real-Time Face Identification via CNN and Boosted Hashing Forest

Yury Vizilter, Vladimir Gorbatsevich, Andrey Vorotnikov, Nikita Kostromov
State Research Institute of Aviation Systems (GosNIIAS), Moscow, Russia
viz@gosniias.ru, gvs@gosniias.ru, vorotnikov@gosniias.ru, nikita-kostromov@yandex.ru

Abstract

The family of real-time face representations is obtained via Convolutional Network with Hashing Forest (CNHF). We learn the CNN, then transform CNN to the multiple convolution architecture and finally learn the output hashing transform via new Boosted Hashing Forest (BHF) technique. This BHF generalizes the Boosted SSC approach for hashing learning with joint optimization of face verification and identification. CNHF is trained on CASIA-WebFace dataset and evaluated on LFW dataset. We code the output of single CNN with 97% on LFW. For Hamming embedding we get CBHF-200 bit (25 byte) code with 96.3% and 2000-bit code with 98.14% on LFW. CNHF with 2000×7-bit hashing trees achieves 93% rank-1 on LFW relative to basic CNN 89.9% rank-1. CNHF generates templates at the rate of 40+ fps with CPU Core i7 and 120+ fps with GPU GeForce GTX 650.

1. Introduction

Various face recognition applications presume different priorities of template size, template generation speed, template matching speed and recognition rates. So, the unified real-time face identification task requires constructing the family of face representations, which provides the flexible balancing of these main properties. We know that the fastest search in a base is provided by binary templates with Hamming distance ([1, 7-10, 12, 14, 18, 20, 21, 30, 34]). On the other hand, the best face recognition rates are achieved by deep convolutional neural networks (CNN) with non-binary face representations ([3, 5, 23-25, 27, 29, 31, 35]). These approaches can be fused in the special CNN architecture with binary output layer, which we refer as Convolutional Network with Hashing Layer (CNHL). The most promising CNHL is described in [6], where CNN and hashing layer are learned together via back propagation technique. But now we need the family of face representations, which continuously varies from small Hamming codes to coded features with larger size, better metrics and higher recognition rates. So, in this paper we propose to combine the CNN and additional hashing transform based on Hashing Forest (HF). Our HF forms the vector of features coded by binary trees. HF with different depth of trees and different coding objectives allows obtaining the family of face representations based on the same CNN. We refer such CNN+HF architecture as Convolutional Network with Hashing Forrest (CNHF). In case of 1-bit coding trees CNHF degrades to CNHL and provides the Hamming embedding.

The architecture of our CNHF is based on the Max-Feature-Map (MFM) CNN architecture proposed by Xiang Wu [31]. For real-time implementation we accelerate our CNN via transforming to the multiple convolution architecture.

We propose the new Boosted Hashing Forest (BHF) technique, which generalizes the Boosted Similarity Sensitive Coding (Boosted SSC) [20, 21] for discriminative data coding by forest hashing with direct optimization of objective function and given properties of coded feature space. We also introduce and implement the new biometric-specific objective function for joint optimization of face verification and identification.

Proposed CNHF face representations are trained on CASIA-WebFace dataset and evaluated on LFW dataset. Our experiments demonstrate both compact binary face representations and increasing of face verification and identification rates. In the Hamming embedding task BHF essentially outperforms the original Boosted SSC. Our CNHF 200 bit (25 byte) hash achieves 96.3% on LFW with 70-time gain in a matching speed. CNHF 2000 bit hash provides 98.14% on LFW. CNHF with 2000×7-bit hashing trees achieves 93% rank-1 on LFW relative to basic CNN 89.9% rank-1.

The remainder of this paper is organized as follows. Section 2 briefly describes the related work. Section 3 describes the architecture and learning of our CNHF with multiple convolution layers. Section 4 contains the outline of proposed BHF technique and its implementation for face hashing. Experimental results are presented in Section 5. Conclusion and discussion are presented in Section 6.

2. Related work

A lot of face representation techniques were proposed ([4, 15, 26]), but all state-of-the-art results are obtained now via deep CNN. One can learn CNN for multi-class

face identification with classes corresponding to persons ([27, 35]), or learn the similarity metric by training two identical CNNs (Siamese Architecture ([5, 29]), or combine these approaches ([23, 24]). Best modern results on LFW are obtained by ensembles of deep nets learned on different parts (patches) of face ([13, 23, 24]). Nevertheless, some single nets can be efficient enough with essentially lower computational cost [3, 31]. Most frequently the CNN-based face representation is formed as an output of top hidden layer [5, 23, 27, 29, 31, 35]. Sometimes the PCA is applied for size reduction [23, 25]. The L2-distance [4, 29] or cosine similarity [23, 27, 31] are of use for matching of face representations.

Binary hashing means the assigning of binary code to each input feature vector. The review of classical hashing techniques is presented in [9]. The simplest binary hashing idea is to use some dimensionality reduction transform and then apply some quantization technique. The optimization-based hashing approach presumes the similarity-driven data embedding into the Hamming space. In [7] the similarity search is proposed based on linear binary coders and vectors of weights obtained by random rotations. The Iterative Quantization (ITQ) technique [8] considers the hashing problem as a search of rotation, which minimizes the quantization error. Kernel-Based Supervised Hashing (KSH) [14] utilizes a kernel formulation for the target hash functions. The affinity-preserving algorithm [10] performs k-means clustering and learns the binary indices of the quantized cells. The manifold hashing techniques follow the ideas of manifold learning. The Spectral Hashing [30] relaxes the hashing problem in the manner of Laplacian Eigenmaps [1]. Topology Preserving Hashing (TPH) [34] perfroms the Hamming embedding with additional preserving the neighbor ranks. Locally Linear Hashing (LLH) [12] presumes both preserving distances and reconstructing the locally linear structures. The Semantic Hashing (SH) [18] solves the hashing problem with the use of Restricted Boltzmann Machines (RBM). Boosted Similarity Sensitive Coding (Boosted SSC) proposed by Shaknarovich, Voila and Darrell [20, 21] performs the sequential bit-by-bit growing of the hash code with reweighting of samples in the manner of AdaBoost and forming the weighted Hamming space.

The idea of binary face coding based on deep learning is well implemented in [6]. The CNN and hashing layer are learned together via back propagation technique, and 32-bit binary face representation is generated with 91% verification on LFW. Unfortunately, the direct optimization of more complex face coding criterions is not available in this one-step CNHL learning framework. In particular, it cannot provide the immediate optimization of Cumulative Matching Curve (CMC). Due to this we implement the two-step CNHF learning procedure: learning basic CNN first and hashing transform second.

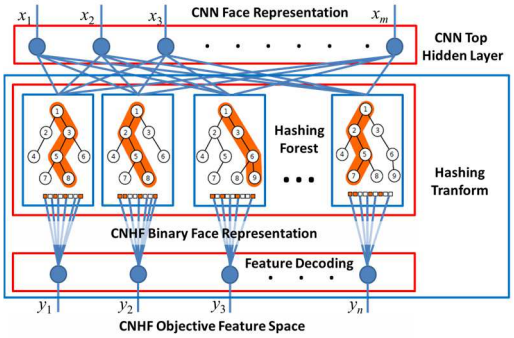


Fig.1. Architecture of CNHF: CNN + Hashing Transform based on Hashing Forest.

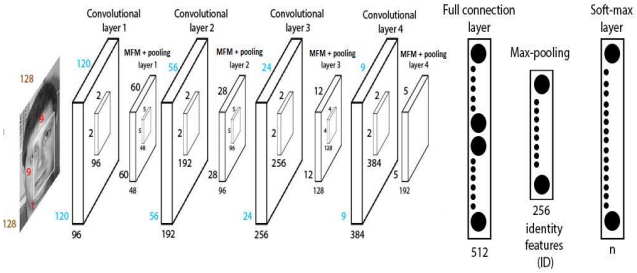


Fig.2. Architecture of source MFM deep net [24]

Our hashing transform is based on hashing forest. Look at some previous forest hashing techniques. Qiu, Sapiro, and Bronstein [17] propose the random forest semantic hashing scheme with information-theoretic code aggregation for large-scale data retrieval. The feature induction based on random forest for learning regression and multi-label classification is proposed by Vens and Costa [28]. Yu and Yuan [33] implement a forest hashing with special order-sensitive Hamming distance. The forest hashing by Springer et al. [22] combines *kd*-trees with hashing technique. The Boosted Random Forest algorithm proposed by Mishina, Tsuchiya and Fujiyoshi [16] is out of the binary hashing topic. Our approach performs the feature space coding via boosted forest hashing in the manner of Boosted SSC with optimizing of task-specific objective function. So, we mainly consider our BHF technique as a generalization of Boosted SSC.

3. CNHF with multiple convolution CNN

Our CNHF contains the basic deep CNN and additional hashing transform based on Hashing Forrest (HF). This hashing forest forms the output CNHF binary face representation, which semantically corresponds to some objective vector of features coded by these binary trees (Fig.1). For obtaining the family of optimized face representations based on the same CNN we use the two-step CNHF learning procedure. At the first step the CNN

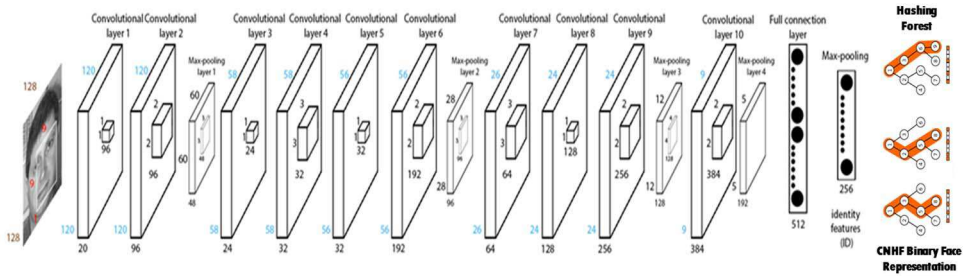


Fig.3. Architecture of CNHF based on MFM net with multiple convolutions.

is formed and trained for multi-class face identification. At the second step the hashing transform is trained for combined face verification and identification.

We start from learning the source CNN with softmax output layer for face identification. Then we transform its convolution layers to the multiple convolution form. Finally we cut the output softmax layer and use the activations of top hidden layer as a basic face representation for further hashing. In this paper we use the Max-Feature-Map (MFM) CNN architecture proposed by Xiang Wu [31]. It is based on the Max-Feature-Map activation function instead of ReLU. [31] demonstrates that Max-Feature-Map can get the compact and discriminative feature vectors. The source network architecture contains 4 convolutional layers, 4 layers of pooling + MFM pooling, 1 fully connected layer and the sofmax layer (Fig.2). Following the approach of Xiang Wu [31] we start from learning this source MFM deep net for multi-class face identification with classes corresponding to persons in the manner [25, 31] using the back-propagation technique. We accelerate our basic CNN via transforming to the multiple convolution architecture. The each convolutional layer is substituted by the superposition of some (2-4) simpler convolutional layers. Such structure allows essentially decreasing the number of multiplication operations in calculation of network output values.

After these simplifying substitutions, the transformed CNN is trained again for multi-class face identification with classes corresponding to persons in the manner [25, 31] using the back-propagation technique. Finally the output soft-max layer of transformed MFM net is replaced by hashing forest, and we obtain the CNHF based on MFM with multiple convolution layers (Fig.3). In result our CNHF contains 10 convolutional layers, 4 layers of MFM+pooling, fully-connected layer and hashing forest. This CNHF generates face templates at the rate of 40+ fps with CPU Core i7 and 120+ fps with GPU GeForce GTX 650.

4. Learning face representation via boosted hashing forest

4.1. Boosted SSC, Forest Hashing and Boosted Hashing Forest

We learn our hashing transform via the new Boosted Hashing Forest (BHF) technique, which combines the algorithmic structure of Boosted SSC [20, 21] and the binary code structure of forest hashing [16, 17, 22, 28, 33].

Boosted SSC algorithms optimize the performance of L1 distance in the embedding space as a proxy for the pairwise similarity function, which is conveyed by a set of examples of positive (similar) and negative (dissimilar) pairs. The *SSC algorithm* takes pairs labeled by similarity and produces a binary embedding space. The embedding is learned by independent collecting thresholded projections of the input data. The threshold is selected by optimal splitting the projections of negative pairs and non-splitting the projections of positive pairs. *Boosted SSC algorithm* collects the embedding dimensions greedily with adaptive weighting of samples and dimensions in the manner of AdaBoost. *BoostPro algorithm* uses a soft thresholding for gradient-based learning of projections.

The differences of proposed BHF w.r.t. Boosted SSC are the following:

1) BHF performs the binary coding of output feature space, which is not binary in general, but can be binary Hamming, if required.

2) BHF performs the direct optimization of any given objective function of output features.

3) BHF learns the objective-driven data projections via RANSAC algorithm without gradient-based optimization.

4) BHF performs the recursive coding by binary trees and forms the hashing forest, while Boosted SSC performs the iterative feature coding and forms hashing vector.

5) BHF performs the adaptive reweighting of training pairs based on their contribution to the objective function, unlike the AdaBoost-style reweighting of Boosted SSC.

6) Boosted SSC forms the weighted Hamming space. Our BHF forms the any given metric space, including non-weighted Hamming space for fastest data search.

The main differences of proposed BHF w.r.t. other forest hashing techniques: we obtain the hashing forest via RANSAC projections and boosting process in the manner of Boosted SSC; we optimize the task-specific objective function in the coded feature space, but not the similarity in the binary code space.

BHF implementation for face recognition has some additional original features: new biometric-specific objective function with joint optimization of face verification and identification; selection and processing of subvectors of the input feature vector; creation of ensemble of independent hash codes for overcoming the limitations of greedy learning. In the next subsections we describe our BHF algorithms in detail.

4.2. BHF: Objective-driven Recurrent Coding

Let the training set $X = \{\mathbf{x}_i \in R^m\}_{i=1,\dots,N}$ contains N objects described by m-dimensional feature vectors. Map X to the n-dimensional binary space: $X = \{\mathbf{x}_i \in R^m\}_{i=1,\dots,N} \to B = \{\mathbf{b}_i \in \{0,1\}^n\}_{i=1,\dots,N}$. This mapping is an *n-bit coder*:

$$\mathbf{h}(\mathbf{x}): \mathbf{x} \in R^m \to \mathbf{b} \in \{0,1\}^n \quad (1)$$

The elementary coder is called the 1-*bit hashing function*:

$$h(\mathbf{x}): \mathbf{x} \in R^m \to b \in \{0,1\} \quad (2)$$

Let some *objective function* (coding criterion) is given and required to be minimized:

$$\mathcal{J}(X, \mathbf{h}) \to min(\mathbf{h}). \quad (3)$$

Denote $\mathbf{h}^{(k)}(\mathbf{x}) = (h^{(1)}(\mathbf{x}),\dots,h^{(k)}(\mathbf{x}))$. The operation of coders concatenation is $\mathbf{h}^{(k)}(\mathbf{x}) := (\mathbf{h}^{(k-1)}(\mathbf{x}), h^{(k)}(\mathbf{x}))$. The Greedy Objective-driven Recurrent Coding (Greedy ORC) algorithm (Algorithm 1) sequentially forms the bits of our coder in a recurrent manner: $h^{(k)}(\mathbf{x}) = h^{(k)}(\mathbf{x}, \mathbf{h}^{(k-1)})$. The proper procedure for learning the each k-th bit is described in the next subsections.

4.3. BHF: Learning elementary projection via RANSAC algorithm

At the k-th step of coder growing

$$\mathcal{J}(X, \mathbf{h}^{(k)}) = \mathcal{J}(X, \mathbf{h}^{(k-1)}, h^{(k)}) \to min\{h^{(k)} \in \mathbf{H}\}, \quad (4)$$

where $\mathbf{H}$ is a class of coders. Consider the class of elementary coders based on thresholded linear projections

$$h(\mathbf{w}, t, \mathbf{x}) = sgn(\textstyle\sum_{k=1,\dots,m} w_k x_k + t), \quad (5)$$

where $\mathbf{w}$ – vector of weights, t – threshold of hashing function, $sgn(u) = \{1, \text{if } u > 0;\ 0 \text{ - otherwise}\}$. In case of (5) function (4) takes the form

Algorithm 1: Greedy ORC

Input data: X, $\mathcal{J}$, n_{ORC}.
Output data: $\mathbf{h}(\mathbf{x})$: $\mathbf{x} \in R^m \to \mathbf{y} \in \{0,1\}^{n_{ORC}}$, $\mathbf{h}(\mathbf{x}) \in \mathbf{H}$.
Initialization:
  Step 0. k:=0; $\mathbf{h}^{(k)} := (\,)$.
Repeat iterations:
  k:= k+1;
  Learn k-th elementary coder:
    $h^{(k)}(\mathbf{x}, \mathbf{h}^{(k-1)})$:= Learn1BitHash($\mathcal{J}$, X, $\mathbf{h}^{(k-1)}$);
  Add k-th elementary coder to the hashing function:
    $\mathbf{h}^{(k)}(\mathbf{x}) := (\mathbf{h}^{(k-1)}(\mathbf{x}), h^{(k)}(\mathbf{x}, \mathbf{h}^{(k-1)}))$;
while $k < n_{ORC}$. // *stop if the given size of coder is got*

Algorithm 2: RANSAC Learn1ProjectionHash

Input data: $\mathcal{J}$, X, $\mathbf{h}^{(k-1)}$, k_{RANSAC}.
Output data: $h(\mathbf{w}, t, \mathbf{x})$.
Initialization:
  Step 0. k:=0; $\mathcal{J}_{max}$:=-∞.
Repeat iterations:
  k:= k+1;
  Step 1. Take the random dissimilar pair $(\mathbf{x}_i, \mathbf{x}_j)$ in X.
  Step 2. Get vector $\overrightarrow{(\mathbf{x}_i, \mathbf{x}_j)}$ as a vector of hyperplane direction: $\mathbf{w}_k := \mathbf{x}_j - \mathbf{x}_i$.
  Step 3. Calculate the threshold t_k minimizing $\mathcal{J}$ (6) by t with $\mathbf{w} = \mathbf{w}_k$: $t_k := \text{argmin}_t\, \mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}_k, t)$.
  Step 4. If $\mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}_k, t_k) > \mathcal{J}_{max}$, then
    $\mathcal{J}_{max} := \mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}_k, t_k)$; $\mathbf{w} := \mathbf{w}_k$; $t := t_k$.
while $k < k_{RANSAC}$. // *stop if the given number of RANSAC iterations is achieved*

Algorithm 3: Boosted Hashing Forest

Input data: X, $\mathcal{J}$, n_{ORC}, n_{BHF}.
Output data: $\mathbf{h}(\mathbf{x})$: $\mathbf{x} \in R^m \to \mathbf{y} \in \{0,1\}^n$.
Initialization:
  l:=0; $\mathbf{h}^{[1,0]}$:= ( ).
Repeat iterations:
  l:= l+1;
  Form the objective as a function of l-th coding tree:
    $\mathcal{J}^{[l]}(X, \mathbf{h}^{[l,l]}) = \mathcal{J}(X, \mathbf{h}^{[1,l-1]}, \mathbf{h}^{[l,l]})$;
  Learn l-th coding tree:
    $\mathbf{h}^{[l,l]}$:= GreedyORC($\mathcal{J}^{[l]}$, X, n_{ORC});
  Add l-th coding tree to the hashing forest:
    $\mathbf{h}^{[1,l]}(\mathbf{x}) := (\mathbf{h}^{[1,l-1]}(\mathbf{x}), \mathbf{h}^{[l,l]}(\mathbf{x}))$;
while $l < n_{ORC}$. // *stop if the given size of coder is got*

$$\mathcal{J}(X, \mathbf{h}^{(k-1)}, h^{(k)}) = \mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}, t) \to min\{\mathbf{w} \in R^m, t \in R\}. \quad (6)$$

We use the RANSAC algorithm for approximate solving (6). RANSAC hypotheses about $\mathbf{w}$ parameters are generated based on the random choice of dissimilar pairs in a training set (Algorithm 2). The selection of threshold at the step 3 is performed in the manner of Boosted SSC

"ThresholdRate" algorithm [20]. For the fixed hypothesis $\mathbf{w}=\mathbf{w}_k$, we arrange the projections $t^{(k)}_i = (\mathbf{x}_i,\mathbf{w}_k)$ and test them (in a linear time) as possible threshold values via calculating the $\mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}_k, t^{(k)}_i)$.

4.4. BHF: Boosted Hashing Forest

Our Learn1BitHash procedure (see Algorithm 1) contains the recursive call of Learn1ProjectionHash procedure (Algorithm 2). Consider the tessellation of X by n-bit coder: $\mathbf{X}_B = \{X_\mathbf{b},\ \mathbf{b}\in\{0,1\}^n\}$, $X_\mathbf{b} = \{\mathbf{x}\in X:\ \mathbf{h}(\mathbf{x})=\mathbf{b}\}$, $X = \cup_{\mathbf{b}\in\{0,1\}^n} X_\mathbf{b}$. The process of recursive coding is a dichotomy splitting of training set with finding the optimized elementary coder for each subset at each level of tessellation. So, the recursive coder for k-th bit

$$h^{(k)}(\mathbf{x},\ \mathbf{h}^{(k-1)}) = h(\mathbf{w}(\mathbf{h}^{(k-1)}(\mathbf{x})),\ t(\mathbf{h}^{(k-1)}(\mathbf{x})),\ \mathbf{x}),$$

is a combination of $2^{(k-1)}$ thresholded projections:

$$h^{(k)}(\mathbf{x},\ \mathbf{h}^{(k-1)}) = \text{Learn1BitHash}(\mathcal{J},\ X,\ \mathbf{h}^{(k-1)}) =$$
$$\{\text{Learn1ProjectionHash}(\mathcal{J},\ X(\mathbf{h}^{(k-1)},\mathbf{b}),\ \mathbf{h}^{(k-1)}),\ \mathbf{b}\in\{0,1\}^{(k-1)}\}.$$

Such recursive n-bit coder $\mathbf{h}(\mathbf{x})$ is a *tree* of thresholded projections, which has much more recognition power relative to the n-bit sequence of thresholded projections.

We know that one coding tree cannot provide the fine recognition rate. Besides, the number of projections in a tree grows exponentially with tree depth. So, the training set of some fixed size allows learning the trees with some limited depth only. Due to this, we form the hashing forest via the boosting of hashing trees with optimization of joint objective function for all trees. We call such approach as Boosted Hashing Forest (BHF) (Algorithm 3).

Here we use the following notation: $n_{ORC} = p$ is a depth of coding tree; $n_{BHF} = n/p$ is a number of trees; $\mathbf{h}^{[1,l]} = (h^{(1)}(\mathbf{x}),\ldots,h^{(lp)}(\mathbf{x}))$, $\mathbf{h}^{[1,l-1]} = (h^{(1)}(\mathbf{x}),\ldots,h^{(lp-p)}(\mathbf{x}))$, $\mathbf{h}^{[l,l]} = (h^{(lp-p+1)}(\mathbf{x}),\ldots,h^{(lp)}(\mathbf{x}))$.

4.5. BHF: Hashing forest as a metric space

We call the metric space (Y, d_Y) with d_Y: $Y\times Y\to R^+$ as *n-bit binary coded*, if the each $y\in Y$ corresponds to unique $\mathbf{b}\in\{0,1\}^n$, and two decoding functions are given: *feature decoder* $f_y(\mathbf{b})$: $\{0,1\}^n \to Y$ and *distance decoder* $f_d(\mathbf{b}_1,\mathbf{b}_2)$: $\{0,1\}^n\times\{0,1\}^n \to R^+$, $f_d(\mathbf{b}_1,\mathbf{b}_2) = d_Y(f_y(\mathbf{b}_1),\ f_y(\mathbf{b}_2))$. This allows define the *distance-based objective function* (DBOF) for coder $\mathbf{h}(\mathbf{x})$ of the form:

$$\mathcal{J}(X,\mathbf{h}) \to min(\mathbf{h}) \Leftrightarrow \mathcal{J}(D_Y) \to min(D_Y), \qquad (7)$$
$$D_Y = \{d_{ij} = f_d(\mathbf{h}(\mathbf{x}_i),\ \mathbf{h}(\mathbf{x}_j)),\ \mathbf{x}_i,\mathbf{x}_j\in X,\ \mathbf{h}(\mathbf{x})\in\mathbf{H}\}_{i,j=1,\ldots,N}.$$

Such objective function depends on the set of coded distances d_{ij} only. In our current implementation of BHF we match p-bit binary trees via the *search index distance*. It is a geodesic distance between codes as corresponding leaves on a coding tree:

$$d_T(y_1,y_2) = f_{dT}(\mathbf{b}_1,\mathbf{b}_2) =$$
$$= 2\sum_{k=1,\ldots,p}(1-\prod_{l=1,\ldots,k}(1-|b_1^{(l)}-b_2^{(l)}|)).$$

Finally, we form a matching distance for total n-dimensional forest containing $q = n/p$ trees as a sum of distances between individual p-bit trees:

$$d_{ij} = \sum_{l=1,\ldots,q} f_{dT}(\mathbf{h}^{[l,l]}(\mathbf{x}_i),\ \mathbf{h}^{[l,l]}(\mathbf{x}_j)).$$

4.6. BHF: Objective function for face verification and identification

Let the similarity function s describes positive (authentic) and negative (imposter) pairs:

$$s_{ij} = \begin{cases} 1, \text{if class}(\mathbf{x_i}) = \text{class}(\mathbf{x_j}), \\ 0, \text{otherwise.} \end{cases} \qquad (8)$$

The "ideal" distance for k-bit binary code, is

$$g^{(k)}{}_{ij} = \begin{cases} 0, & \text{if } s_{ij} = 1, \\ d_{max}(k), & \text{otherwise,} \end{cases} \qquad (9)$$

where $d_{max}(k)$ is a maximal possible distance. So, the *distance supervision objective function* can be formed as

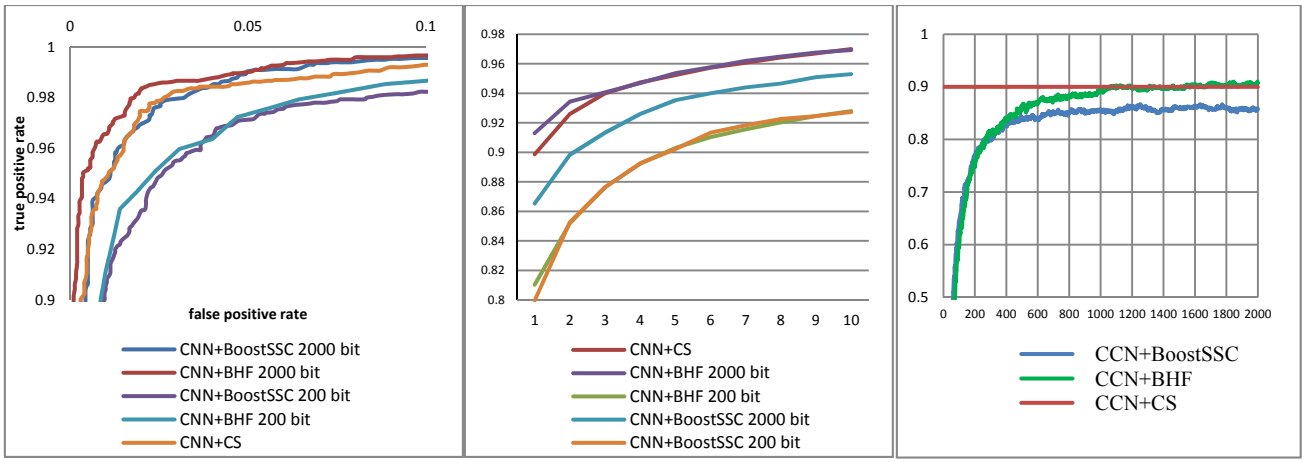
$$\mathcal{J}_{Dist}(D_Y) = \sum_{i=1,\ldots,N}\sum_{j=1,\ldots,N} v_{ij}\,(d_{ij}-g_{ij})^2 \to$$
$$\to min(D_Y = \{d_{ij}\}_{i,j=1,\ldots,N}), \qquad (10)$$

where v_{ij} are the different weights for authentic and imposter pairs. This objective function (10) controls the verification performance (FAR and FRR).

In the identification-targeted biometric applications we need to control both distances and ordering of distances. Let $d^1{}_k = \max_l\{d_{kl}:\ s_{kl} = 1\}$ is a distance to the most far authentic and $d^0{}_k = \min_l\{d_{kl}:\ s_{kl} = 0\}$ is a distance to the closest imposter for the query $\mathbf{h}(\mathbf{x}_k)$. Then the ordering error e_{ij} for a pair $(\mathbf{x}_i,\mathbf{x}_j)$ can be expressed as

$$e_{ij} = \begin{cases} 1, & \text{if } (s_{ij} = 0\ and\ h_{ij} < \max(d_i^1, d_j^1)) \\ & \text{or } (s_{ij} = 1\ and\ h_{ij} > \min(d_i^0, d_j^0)), \\ 0, & \text{otherwise} \end{cases} \qquad (11)$$

The ordering error occurs if imposter is closer than authentic or authentic is more far than imposter. So, the



(a) (b) (c)

Fig.4. ROC curves (a), CMC curves (b) and identification performance (rank 1) (c) on LFW relative to the size of biometric template in bits for proposed BHF(CNN+BHF) and original Boosted SSC(CNN +BoostSSC) and best basic CNN solution without hashing - CNN + Last hidden layer + cosine similarity (CNN+CS)

distance order supervision objective function can be formed as

$$\mathcal{J}_{Ord}(D_Y) = \sum_{i=1,\dots,N}\sum_{j=1,\dots,N} v_{ij}\,(d_{ij}-g_{ij})^2\,e_{ij} \rightarrow \\ \rightarrow min(D_Y = \{d_{ij}\}_{i,j=1,\dots,N}). \qquad (12)$$

Here we penalize the difference between d_{ij} and objective distance g_{ij} like in (10), but only in case that the ordering error (11) occurs for this pair. So, criterion (12) directly controls the face identification characteristics (CMC).

Finally, for obtaining both verification and identification we combine the (10) and (12) resulting in

$$\mathcal{J}(D_Y) = \alpha\,\mathcal{J}_{Dist}(D_Y) + (1-\alpha)\,\mathcal{J}_{Ord}(D_Y) = \\ = \sum_{i=1,\dots,N}\sum_{j=1,\dots,N} v_{ij}\,(d_{ij}-g_{ij})^2\,(e_{ij}+\alpha(1-e_{ij})) \rightarrow \\ \rightarrow min(D_Y = \{d_{ij}\}_{i,j=1,\dots,N}), \qquad (13)$$

where $\alpha \in [0,1]$ is a tuning parameter.

4.5. BHF implementation for learning face representation

For enhancement of our face representation learning we use some additional semi-heuristic modifications of described scheme. The goal distance (9) is modified:

$$g^{(k)}{}_{ij} = \begin{cases} 0, & \text{if } s_{ij} = 1, \\ m^{(k-1)}{}_1 + 3\sigma^{(k-1)}{}_1, & \text{otherwise,} \end{cases} \qquad (14)$$

where $m^{(k-1)}{}_1$ and $\sigma^{(k-1)}{}_1$ are the mean value and standard deviation of authentic coded distances. Such goal distance (14) excludes the penalizing of imposter pairs, which could not be treated as authentic. In (13) we use the adaptive weighting of pairs at each k-th step of boosting:

$$v^{(k)}{}_{ij} = \begin{cases} \gamma/a^{(k)}, & \text{if } s_{ij} = 1, \\ 1/b^{(k)}, & \text{otherwise,} \end{cases} \qquad (15)$$

$$a^{(k)} = \sum_{i=1,\dots,N}\sum_{j=1,\dots,N} s_{ij}\,(d_{ij}-g_{ij})^2\,(e_{ij}+\alpha(1-e_{ij})),$$
$$b^{(k)} = \sum_{i=1,\dots,N}\sum_{j=1,\dots,N} (1-s_{ij})\,(d_{ij}-g_{ij})^2\,(e_{ij}+\alpha(1-e_{ij})),$$

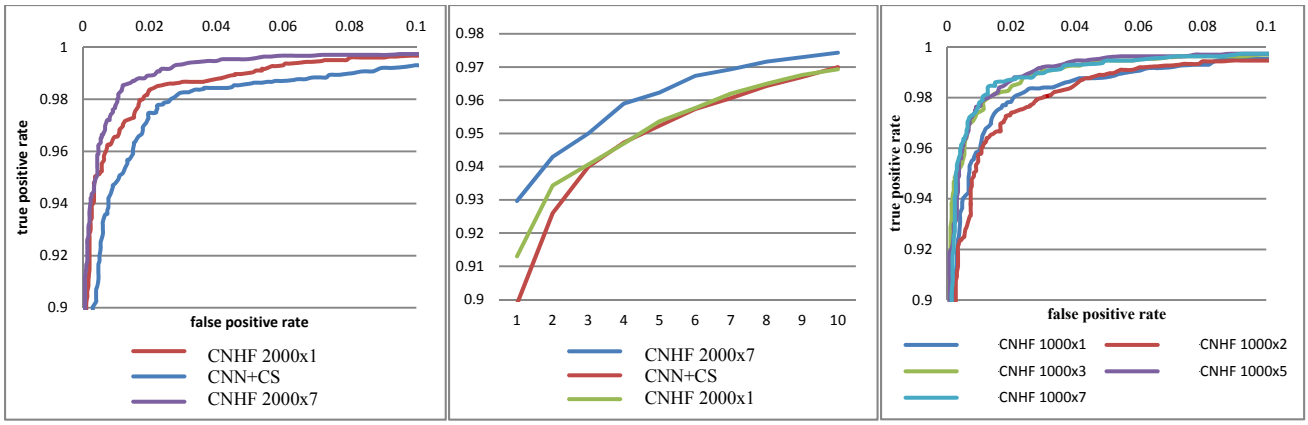
where $a^{(k)}$ and $b^{(k)}$ provide the basic equal weight for all authentic and imposter pairs, and tuning parameter $\gamma>1$ gives the slightly larger weights to authentic pairs.

We split the input m-dimensional feature vector to the set of independently coded subvectors with fixed sizes from the set $\mathbf{m} = \{m_{\min},\dots,m_{\max}\}$. At the each step of boosting we get the subvector with corresponding BHF elementary coder providing the best contribution to the objective function. The output binary vector of size n consists of some independently grown parts of size $n_{BHF}<n$. Such learning strategy prevents the premature saturation of objective function.

So, our binary face hashing is implemented with the following set of free parameters: $\mathbf{m}$, n_{ORC}, n_{BHF}, k_{RANSAC}, α and γ. The type of coded metrics is a free parameter of our approach too.

5. Experiments

In this section we describe our methodology for learning and testing CNHF, report our results in Hamming embedding task, compare proposed BHF to original Boosted SSC, explore the CNHF performance w.r.t. depth of coding trees and compare CNHL and CNHF to best methods on LFW. We test the verification accuracy by the standard LFW unrestricted with outside labeled data protocol. Our CMC and rank-1 tests follow the methodology described in [2].



(a) (b) (c)

Fig.5. ROC (a) and CMC (b) curves for CNN+CS, CNHF-2000×1 and CNHF-2000×7; ROC curves for CNHF-1000×*p*-bit trees

Table 1. Verification accuracy on LFW, code size and matching speed of CNN and CNHL

Solution	Accuracy	Template size	Matches in sec
CNN+L2	0.947	8192 bit	2713222
CNN+BHF-200×1	0.963	**200 bit**	**194986071**
CNN+CS	0.975	8192 bit	2787632
CNN+BHF-2000×1	**0.9814**	**2000 bit**	**27855153**

Table 2. Verification accuracy on LFW.

Method	Accuracy
WebFace [25]	0.9613
CNHL-200×1	**0.963±0.00494**
DeepFace-ensemble[21]	0.9730±0.0025
DeepID[19]	0.9745± 0.0026
MFM Net[24]	0.9777
CNHL-2000×1	**0.9814**
CNHF-2000×7	**0.9859**
DeepID2[17]	0.9915 ± 0.0013
DeepID3[18]	0.9953 ± 0.0010
Baidu[11]	0.9977 ± 0.0006

5.1 Methodology: learning and testing CNHF

The basic CNN is trained on CASIA-WebFace dataset. Face images are aligned by rotation of eye points to horizontal position with fixed eye-to-eye distance and crop to 128x128 size. The open source deep learning framework Caffe (http://caffe.berkeleyvision.org/) is used for training the basic CNN for multi-class face identification in the manner [25, 31]. The hashing forest is trained on the dataset containing 1000 authentic pairs and correspondingly 999000 imposter pairs of Faces in the Wild images (not from the testing LFW set). Finally, the family of CNHF coders is formed by proposed BHF: Hamming embedding coders 2000×1 bit (250 byte), 200×1 bit (25 byte) and 32×1 bit (4 byte) of size; Hashing forest coders containing 2000 trees with 2-7 bits depth (0.5 – 1.75 Kbyte of size). We used the common setting of BHF parameters: $\mathbf{m} = \{8, 16, 32\}$, $k_{RANSAC} = 0$, $\alpha = 0.25$, $\gamma = 1.1$. But we set n_{BHF}=200 for CNN+BHF-200×1, n_{BHF}=500 for CNN+BHF-2000×1 and n_{BHF}=100 for CNHF-2000×7. Such parameter values are determined experimentally based on the analysis of the speed of identification rate growing w.r.t. number of code bits in the hashing process. The evaluation is performed on the Labeled Faces in the Wild (LFW) dataset. All the images in LFW dataset are processed by the same pipeline as in [11] and normalized to 128x128.

5.2 Hamming embedding: CNHL vs. CNN, BHF vs. Boosted SSC

In this subsection we test our approach in Hamming embedding task, so, CNHF degrades to CNHL. We compare CNHL to basic CNN on LFW via verification accuracy and ROC curve (Table 1 and Fig.4a). The CNN face representation is formed like in [34] as a vector of activations of 256 top hidden layer neurons. The cosine similarity (CNN+CS) and L2-distance (CNN+L2) are applied for matching. CNHL coders 2000 and 200 bit of size are trained by BHF and matched by Hamming distance (CNN+BHF-2000×1 and CNN+BHF-200×1 correspondingly). Our solution CNN+BHF-2000×1 achieves verification accuracy 98.14% on LFW, which outperforms all other CNN-based solutions. Moreover, our 25-byte length solution CNN+BHF-200×1 outperforms CNN+L2. Table 1 additionally demonstrates the gain in template size and matching speed.

We compare CNHL trained by BHF to CNHL trained by original Boosted SSC. Fig.4c demonstrates that proposed BHF essentially outperforms Boosted SSC in identification (rank-1) on LFW for all binary template

sizes. The maximal rank-1 is 0.91 for BHF-2000×1 and 0.865 for BoostSSC-2000×1 (relative to 0,899 for CNN+CS). The ROC graph for CNN+BHF is monotonously better than for CNN+BoostSSC with same template size (Fig.4a). Fig.4b contains the CMC graphs (ranks 1-10), which demonstrate that BHF outperforms BoostSSC with same template size (additionally note that CNN+BHF-2000×1 outperforms CNN+CS).

5.3 CNHF: performance w.r.t. depth of trees

CNHF with 2000 output features formed by 7-bit coding trees (CNHF-2000×7) achieves 98.59% on LFW. The identification result of CNHF-2000×7 is 93% rank-1 on LFW relative to 89.9% rank-1 for CNN+CS. Fig.5c presents the ROC curves for CNHF with different depth coding trees. The forest with 7-bit coding trees is the best by ROC, but 6-bit and 5-bit depth solutions are very close. We suppose that the reason of this result is a limited amount of hashing forest training set. Fig.5a,b demonstrates that CNHF-2000×7 outperforms basic CNN+CS and CNHF-2000×1 both in verification (ROC) and in identification (CMC). So, we can conclude that the adding of hashing forest on the top of CNN allows both generating the compact binary face representation and increasing the face verification and especially identification rates.

5.4 CNHL and CNHF vs. best methods on LFW

We compare our CNHF solutions to state-of-the-art methods (best on LFW) via verification accuracy (Table 2). CNHF-2000×1 outperforms DeepFace-ensemble [30], DeepID [27], WebFace [35] and MFM Net [34]. The DeepID2 [25], DeepID3 [26] and Baidu [14] multi-patch CNNs outperform our CNHF-2000×1 based on single net.

Note that our CNHF-200×1 (25 byte) hash demonstrates 96.3% on LFW. Compare this result to previous best CNHL result [7]. On the one hand, the extreme-short 32-bit binary face representation [7] achieves 91% verification on LFW. Our CNHF 32×1 provides 90% only. On the other hand, face representation [6] requires 1000 bit for achieving the 96% verification on LFW. So, our CNHF-200×1 solution improves this face packing result in 5 times.

The identification result (rank-1) of our real-time coder CNHF-2000×7 is 0.93 on LFW. It is close enough to best reported identification result of essentially deeper and slower multi-patch DeepID3 CNN [24] (0.96 rank-1 on LFW). Baidu [13] declares even better result (0.98 rank-1 on LFW), but they use the training set 1.2 million images of size w.r.t. 400 thousand images in our case.

6. Conclusion and Discussion

We develop the family of CNN-based binary face representations for real-time face identification. Our Convolutional Network with Hashing Forest (CNHF) generates binary face templates at the rate of 40+ fps with CPU Core i7 and 120+ fps with GPU GeForce GTX 650. Our 2000×1-bit face coder provides the compact face coding (250 byte) with simultaneous increasing of verification (98.14%) and identification (91% rank-1) on LFW. Our 200×1-bit face coder provides the 40-time gain in template size and 70-time gain in a matching speed with 1% decreasing of verification accuracy relative to basic CNN (96.3% on LFW). Our CNHF with 2000 output 7-bit coding trees (CNHF-2000×7) achieves 98.59% verification accuracy and 93% rank-1 on LFW (add 3% to rank-1 of basic CNN).

We use the multiple convolution deep network architecture for acceleration of source Max-Feature-Map (MFM) CNN architecture [31]. We propose and implement the new binary hashing technique, which forms the output feature space with given metric properties via joint optimization of face verification and identification. This Boosted Hashing Forest (BHF) technique combines the algorithmic structure of Boosted SSC approach and the binary code structure of forest hashing. Our experiments demonstrate that BHF essentially outperforms the original Boosted SSC in face identification test.

In the future we will try to achieve the better recognition rates via CNHF based on multi-patch CNN, which we can use for non-real-time applications. We will evolve and apply the proposed BHF technique for different data coding and dimension reduction problems (supervised, semi-supervised and unsupervised). Additionally, we will investigate the influence of the output metric space properties in the process of hashing forest learning.

Acknowledgement

This work is supported by grant from Russian Science Foundation (Project No. 16-11-00082).

References

[1] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” *Proc. NIPS 14*, pp. 585– 591, 2001.

[2] L. Best-Rowden, H. Han, C. Otto, B. Klare and A. K. Jain, “Unconstrained face recognition: Identifying a person of interest from a media collection,” *IEEE Trans. Inf. Forens. Security*, vol. 9, no. 12, pp. 2144-2157, 2014.

[3] Z. Cao, Q. Yin, X. Tang and J. Sun. "Face Recognition with Learning-based Descriptor," *Proc. CVPR*, pp. 2707-2714, 2010.

[4] D. Chen, X. Cao, F. Wen and J. Sun, "Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification," *Proc. CVPR*, pp. 3025-3032, 2013.

[5] H. Fan, Z. Cao, Y. Jiang, Q. Yin and C. Doudou, "Learning deep face representation," *arXiv preprint arXiv:1403.2802*, 2014.

[6] H. Fan, M. Yang, Z. Cao, Y. Jiang and Q. Yin, "Learning Compact Face Representation: Packing a Face into an int32," *Proc. ACM Int. Conf. Multimedia*, pp. 933-936, 2014.

[7] A. Gionis, P. Indyk and R. Motwani, "Similarity search in high dimensions via hashing," *Proc. VLDB*, pp. 518-529, 1999.

[8] Y. Gong, S. Lazebnik and A. Gordo, Florent 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, 2012.

[9] K. Grauman and R. Fergus, "Learning binary hash codes for large-scale image search," *in Machine Learning for Computer Vision*, pp. 49-87, Springer, 2013.

[10] K. He, F. Wen and J. Sun, "K-means Hashing: An affinity-preserving quantization method for learning binary compact codes," *Proc. CVPR*, pp. 2938-2945, 2013.

[11] G.-B. Huang, M. Mattar, H. Lee and E. Learned-Miller, "Learning to align from scratch," *Proc. NIPS 25*, 2012.

[12] G. Irie, L. Zhenguo, W. Xiao-Ming and C. Shih-Fu, "Locally linear hashing for extracting non-linear manifolds," *Proc. CVPR*, pp. 2115-2122, 2014.

[13] J. Liu, Y. Deng, T. Bai, Z. Wei and C. Huang, "Targeting ultimate accuracy: face recognition via deep embedding," *arXiv preprint arXiv:1506.07310*, 2015.

[14] W. Liu, J. Wang, R. Ji, Y.-G. Jiang and S.-F. Chang, "Supervised hashing with kernels," *Proc. CVPR*, pp. 2074-2081, 2012.

[15] H.-V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," *Proc. ACCV*, pp. 709-720, 2010.

[16] Y. Mishina, M. Tsuchiya and H. Fujiyoshi, "Boosted Random Forest," *IEICE Trans.*, vol. E98-D, no. 9, pp. 1630-1636, 2015.

[17] Q. Qiu, G. Sapiro and A. Bronstein, "Random Forests Can Hash," *arXiv preprint arXiv:1412.5083*, 2014.

[18] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, Pages 969–978, 2009.

[19] F. Schroff, D. Kalenichenko and James Philbin, "FaceNet: A unified embedding for face recognition and clustering," *Proc. CVPR*, pp. 815-823, 2015.

[20] G. Shakhnarovich, "Learning task-specific similarity," PhD thesis, Dept. of Elect. Eng. and Comput. Sci., MIT, Cambridge, MA, 2005.

[21] G. Shakhnarovich, P. Viola and T. Darrell, "Fast pose estimation with parameter sensitive hashing," *Proc. Comput. Vision*, vol. 2, pp. 750-757, Oct. 2003.

[22] J. Springer, X. Xin, Z. Li, J. Watt and A. Katsaggelos, "Forest hashing: Expediting large scale image retrieval," *Proc. ICASSP*, pp. 1681-1684, May 2013.

[23] Y. Sun, X. Wang and X. Tang, "Deep learning face representation by joint identification-verification," *Proc. NIPS 27*, 2014.

[24] Y. Sun, X. Wang and X. Tang, "DeepID3: Face recognition with very deep neural networks," *arXiv preprint arXiv:1502.00873*, 2015.

[25] Y. Sun, X. Wang and X. Tang, "Deep learning face representation from predicting 10,000 classes," *Proc. CVPR*, pp. 1891-1898, 2014.

[26] Y. Taigman, L. Wolf and T. Hassner, "Multiple one-shots for utilizing class label information," *Proc. BMVC*, 2009.

[27] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "DeepFace: closing the gap to human-level performance in face verification," *Proc. CVPR*, pp. 1701-1708, 2014.

[28] C. Vens, F. Costa, "Random Forest Based Feature Induction," *Proc. ICDM*, pp. 744-753, 2011.

[29] W. Wang, J. Yang, J. Xiao, S. Li and D. Zhou, "Face recognition based on deep learning," *Proc. HCC*, vol. 8944, pp. 812-820, Mar. 2015.

[30] Y. Weiss, A. Torralba and R. Fergus, "Spectral Hashing", *Proc. NIPS 21*, 2008.

[31] X. Wu, "Learning robust deep face representation," *arXiv preprint arXiv:1507.04844*, 2015.

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

[33] G. Yu, J. Yuan, "Scalable forest hashing for fast similarity search," *Proc. ICME*, pp. 1-6, 2014.

[34] L. Zhang, Y. Zhang, X. Gu, J. Tang and Q. Tian, "Topology preserving hashing for similarity search," *Proc. ACM Int. Conf. Multimedia*, pp. 123-132, 2013.

[35] E. Zhou, Z. Cao and Q. Yin, "Naive-deep face recognition: Touching the limit of LFW benchmark or not?" *arXiv preprint arXiv:1501.04690*, 2015.