MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Improving face verification and person re-identification accuracy using hyperplane similarity

Jones, M.J.; Kobori, H.

TR2017-155 October 2017

Abstract

The standard framework for using a convolutional neural network (CNN) for face verification is to compare the feature vectors taken from the penultimate network layer of a CNN trained to classify the identity of an input face using a softmax loss over identities. Feature vectors are typically compared using the simple L2 distance. We demonstrate that the L2 distance is not the best distance to use in this scenario, and propose the hyperplane similarity as a more appropriate similarity function that is derived from the softmax loss function used to train the network. We demonstrate that hyperplane similarity improves verification results especially for low false acceptance rates which are usually the most important operating regimes for real applications. We also propose a fast algorithm for finding the separating hyperplanes needed to compute hyperplane similarity.

IEEE Workshop on Analysis and Modeling of Faces and Gestures (at ICCV)

Improving face verification and person re-identification accuracy using hyperplane similarity

Hiroko Kobori
Mitsubishi Electric
Kanagawa, Japan 247-8501
Kobori.Hiroko@ct.MitsubishiElectric.com

Michael Jones
Mitsubishi Electric Research Labs
Cambridge, Massachusetts 02139
mjones@merl.com

Abstract

The standard framework for using a convolutional neural network (CNN) for face verification is to compare the feature vectors taken from the penultimate network layer of a CNN trained to classify the identity of an input face using a softmax loss over identities. Feature vectors are typically compared using the simple L_2 distance. We demonstrate that the L_2 distance is not the best distance to use in this scenario, and propose the hyperplane similarity *as a more appropriate similarity function that is derived from the softmax loss function used to train the network. We demonstrate that hyperplane similarity improves verification results especially for low false acceptance rates which are usually the most important operating regimes for real applications. We also propose a fast algorithm for finding the separating hyperplanes needed to compute hyperplane similarity.*

1. Introduction

In recent years a standard deep convolutional neural network (CNN) architecture for face recognition has emerged that achieves excellent accuracy on various difficult test sets [13, 16, 21, 20]. The architecture takes a cropped face image as input and uses a strong baseline CNN such as VGG [18] or ResNet [7] to compute a feature vector followed by a fully connected layer that outputs a vector of length C where C is the number of unique identities in the training set. The network is trained to minimize the softmax loss between the output vector and a one-hot encoding of the correct identity for the input face image. After training, the final fully connected layer plus softmax function that gives the probability of each training identity is discarded since the training identities are not the same as the identities encountered during testing. Instead, the output of the layer before the final fully connected layer is used as a feature vector. Feature vectors for two testing face images are L_2 normalized and compared using a simply L_2 distance (or, equivalently, cosine similarity).

Despite the good results achieved with this basic architecture [16, 18, 13, 23], there is a fundamental mismatch between how the network is trained and how it is used during testing. During testing in which feature vectors are compared using L_2 distance, the assumption is that feature vectors for same-face pairs will be close in feature space while feature vectors for different-face pairs will be farther apart. However, this property is not being optimized during training. The property that is being optimized is that feature vectors for a particular person are linearly separable from feature vectors for all other people [23]. To address this mismatch, we propose to compare feature vectors according to their distance to hyperplanes that separate one person's feature vectors from all other person's feature vectors. This is in accordance with the training loss.

The basic idea is to use feature vectors for a set of negative faces, which are simply faces from a variety of different people, and compute hyperplanes between these faces and each of the two faces being compared. The sum of the margins to these hyperplanes from the feature vectors of the two test faces can then be used in place of L_2 distance. We will call this similarity function *hyperplane similarity*, and define it more formally in section 3. The negative faces can be face images from the training set or a random set of faces collected from the web, for example. The hyperplane similarity allows us to achieve a significant improvement in verification accuracy while using an existing deep network trained for face identification with the simple softmax loss. Another advantage of hyperplane similarity is that it naturally extends to comparing sets of images. We will show test results on the IJB-A face recognition test set [9] which compares sets of images and videos of a person (called a template) to other sets.

While the advantage of the hyperplane similarity idea is accuracy, the main drawback is speed. The straightforward method to estimate hyperplanes at test time is to use a linear support vector machine (SVM) solver, but this is much slower than using L_2 distance. To address this drawback,

we introduce a simple algorithm to compute a separating hyperplane that does not involve SVM optimization. In our experiments, this simple algorithm proves to have accuracy close to the SVM hyperplane with much greater speed.

In the remainder of the paper we will motivate and explain hyperplane similarity, show that it improves accuracy on multiple face verification and person re-identification test sets, and propose an approximation algorithm that computes hyperplanes much faster than SVM while maintaining improved accuracy compared to L_2 distance.

2. Related Work

Recently a number of state-of-the-art papers on face recognition have used the basic framework sketched in the last section, namely a CNN with a fully connected last layer that outputs a probability of each training identity, and is trained using a softmax loss [13, 21]. Various researchers have noted the mismatch between the training criteria and the usage of the penultimate network layer as a feature vector with distance between feature vectors measured with L_2 distance. Most previous work attempts to address this mismatch by proposing a new loss function for training.

One of the best known alternative loss functions is the *triplet loss* [6, 17]. The triplet loss takes an "anchor" face as well as positive and negative example images of the anchor's identity as an input example. The triplet loss attempts to minimize the distance between the anchor and positive feature vectors minus the distance between the anchor and negative feature vectors. In other words, the triplet loss explicitely tries to minimize the distance between same-face pairs while maximizing distance between different-face pairs. One difficulty with this loss is that the number of triples of face images for training becomes very large and some kind of hard-negative mining is needed. Another loss function, known as *contrastive loss* [19, 6], has a similar effect to the triplet loss using a slightly different loss function.

The *center loss* proposed by [23] attempts to minimize the distance between a face's feature vector and the mean feature vector for the class (the set of face images for a particular person). Using center loss plus softmax loss tends to yield clusters of feature vectors for each person that are compact and separable from other identities.

Three other related loss functions, *A-softmax* [11] (for angular softmax), *large-margin softmax* [12] and L_2-*constrained softmax* [14] modify the standard softmax loss function in a way that encourages feature vectors of a particular identity to cluster near each other.

A different way to formulate the face recognition problem is to formulate it as a verification problem in which two face images are given as input and a "Siamese-network" CNN is trained to predict whether the two face images are of the same person or different people [21, 19].

All of these various formulations have their advantages and disadvantages. They represent an alternative to the approach we take which is to use a CNN trained using standard softmax loss, but instead change the distance function used to compare feature vectors.

The only other paper to take a similar approach to ours (to the best of our knowledge) is an arXiv paper by Crosswhite et al. [3] that uses a technique they call template adaptation. While the motivation for their method is different from ours, the technique of learning separating hyperplanes to measure the distance between face images (or two sets of face images) is the same. However, our work differs from theirs in a number of ways. In addition to our alternative justification for this formulation, we also demonstrate that it improves verification accuracy on multiple test sets ([3] only used IJB-A), and propose a fast approximation algorithm for finding separating hyperplanes.

We should also note that various papers have replaced the last fully connected layer of a CNN (trained for an image recognition or classification task) with a support vector machine [8, 22]. However, these are not the same as our hyperplane similarity function. In these previous papers, the SVM performed the same function as the last fully connected layer - mapping a feature vector to one of the training classes. In contrast, the hyperplanes we learn are not used to classify the input as a training class, but instead are used to define a similarity function between feature vectors.

3. SVM Face Verification

In this section, we explain the face verification method using hyperplane similarity and the justification for the algorithm. We also discuss the effect from templates with multiple images.

3.1. Verification algorithm

For the sake of generality, we explain hyperplane similarity using templates, where a template is a set of one or more images of a particular person. This is consistent with the terminology used for the IJB-A test set [9]. Given a pair of templates P and Q, the goal of the verification problem is to correctly classify whether they belong to the same identity or not. The 1:N identification problem is also solved in a similar way by interpreting it as a combination of N verification problems and selecting the identity with the largest similarity score.

To solve the verification problem, we assume a CNN has already been trained in the standard way (as described in the introduction) so that a feature vector is produced by the output of the penultimate network layer. A template, P, can then be represented by the set of feature vectors, F_P, calculated by the CNN for each image in the template. A hyperplane is calculated to discriminate the feature vectors, F_P, from a set of negative feature vectors, F_N, extracted

from a set of negative images, N. N is a set of face images of many different people. Let H_p be the hyperplane that separates F_P and F_N. H_p is defined by a normal vector, $\boldsymbol{w}_p$, with the same dimensionality as a feature vector and a scalar offset, b_p. All feature vectors are L_2 normalized before hyperplane calculation. Similarly, template Q is represented by feature vectors F_Q and hyperplane H_Q.

The hyperplane similarity, $S(P, Q, N)$, between two templates can now be defined.

$$S(P, Q, N) = \frac{1}{2}M(H_P, F_Q) + \frac{1}{2}M(H_Q, F_P), \quad (1)$$

where $M(H_P, F_Q)$ is the margin between the set of feature vectors, F_Q, and the hyperplane, H_P, optimized to separate F_p from F_N. $M(H_P, F_Q)$ is the average of all the margins from the $n \geq 1$ feature vectors in F_Q. More formally,

$$M(H_p, F_Q) = \frac{1}{n}\sum_{i=1}^{n} margin(H_p, F_Q(i)) \quad (2)$$

$$margin(H_p, F_Q(i)) = \boldsymbol{w}_p \cdot F_Q(i) + b_p \quad (3)$$

where $\boldsymbol{w}_p$ is the normal vector for hyperplane H_p, b_p is the offset and $F_Q(i)$ is the i^{th} feature vector in set F_Q. Note that the margin between a feature vector and a hyperplane is the signed distance to the hyperplane, i.e. if a feature vector is on the "negative" side of the hyperplane defined by the normal, the margin will be negative.

3.2. Justification for the Hyperplane Similarity

If $\boldsymbol{f}$ is the feature vector for training example $\boldsymbol{x}$ taken from the CNN layer before the last fully connected layer, the output of the fully connected layer z is

$$\boldsymbol{z} = W \cdot \boldsymbol{f} + \boldsymbol{b} \quad (4)$$

where W is the weight matrix of the last fully connected layer and $\boldsymbol{f}$ is the vector of biases. $\boldsymbol{z}$ is a vector of length C representing, for each of the C training identities, the probability that img $\boldsymbol{x}$ belongs to that identity.

During training, the network parameters are optimized the softmax loss over all training examples is minimized. The softmax loss for training example $\boldsymbol{x}$ is near 0 (minimum) when the element of $\boldsymbol{z}$ corresponding to the correct identity is large (positive) compared to the elements of $\boldsymbol{z}$ corresponding to the wrong identities (which optimally are negative). This will occur if W_i (the weights for the i^{th} output) maps feature vectors of identity i to positive values and feature vectors of all other identities to negative values. In other words, the softmax loss is minimized when the columns of matrix W (along with offsets $\boldsymbol{b}$) are separating hyperplanes for each of the different training identities. This is the motivation for hyperplane similarity, which is to match the criteria that is optimized during training.

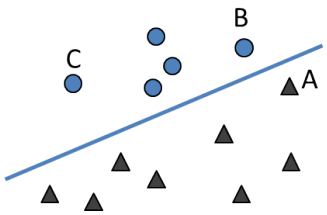

Figure 1. Feature vectors and classification layer weights (=hyperplanes) are trained to maximize the margin between features and the corresponding class hyperplane. When L_2 distance is used as a verification metric, negative pair A-B has smaller distance than positive pair B-C.

It is not guaranteed that the L_2 distance between feature vectors of the same identity is smaller than the distance between different identities. This can be seen in the example shown in Figure 1. Features represented by blue circles are trained to be linearly separable from the features of other classes (represented by black triangles). In this case, all the features and the hyperplane satisfy the training criteria, but the L_2 distance of the negative pair A-B is smaller than the positive pair B-C. For correct verification using L_2 distance, all the features in the same class need to exist in a very tight area, which does not directly match how the CNN is trained.

Face images in the test set contain identities not seen during training and thus separating hyperplanes are not known for them. However, they can be computed given one or more feature vectors for a testing identity along with a set of "negative" feature vectors of other identities.

Multi Image Template Intuitively, the hyperplane can explain a class's feature distribution better when more images are available in the template. To evaluate the improvement gained from the multi-image template, Table 3.2 shows the verification result on IJB-A with different template subsets. The table contains the true acceptance rate (TAR) for given false acceptance rates (FAR). *One media* uses only one media per template where a media is defined as either a single photo or a single video clip. *Averaged* uses the average of all the feature vectors in the template as the only positive example for hyperplane estmation. *Non-averaged* uses all the template feature vectors as positive examples for hyperplane estimation. Two different metrics are compared, L_2 distance and hyperplane similarity (HS). Linear SVM is used for hyperplane estimation.

With both metrics, using averaged features achieved significantly better accuracy compared with using only one feature. Moreover, the hyperplane similarity method achieved a better result with non-averaged features. This result shows that the hyperplane similarity method can utilize the feature distribution information from multiple images for better similarity estimation.

IJB-A Verification (TAR@FAR)			
method	0.1	0.01	0.001
l2 norm (one media)	0.834	0.545	0.287
l2 norm (averaged)	0.959	0.830	0.617
HS (one media)	0.828	0.623	0.421
HS (averaged)	0.971	0.905	0.797
HS (non-averaged)	0.980	0.924	0.830

Table 1. Verification result on IJB-A with different subset of the template.

4. Computation Reduction

To compute the hyperplane that separates features of one identity from others, the straightforward way is to solve linear SVM. However, solving SVM for each verification test is computationally expensive. In this section, we propose a computationally efficient algorithm to estimate a discriminative hyperplane.

4.1. Discriminative Hyperplane Approximation

The intuition behind our fast method for finding a discriminating hyperplane separating a set of positive and negative feature vectors is illustrated in Figure 2. For many distributions of positive and negative feature vectors, the vector pointing from the mean of the negative feature vectors to the mean of the positive feature vectors is normal to a separating hyperplane. Therefore, we set the normal of the separating hyperplane to the difference between the mean positive feature vector and the mean negative feature vector:

$$\mathbf{w} = \frac{1}{p}\sum_{i=1}^{p} F_P(i) - \frac{1}{n}\sum_{j=1}^{n} F_N(j) \tag{5}$$

where $F_P(i)$ is the i^{th} positive feature vector, $F_N(j)$ is the j^{th} negative feature vector, p is the number of positive feature vectors, and n is the number of negative feature vectors.

After we have the normal for the separating hyperplane, we just need the offset which tells the position along the normal that best separates positive from negative feature vectors. This is done by computing the dot product of each positive and negative feature vector with the normal vector. The offset is then set to the average of the minimum positive dot product and the maximum negative dot product.

$$b = \frac{1}{2}\min_i(\mathbf{w}\cdot F_P(i)) + \frac{1}{2}\max_j(\mathbf{w}\cdot F_N(j)) \tag{6}$$

This corresponds to the midpoint between the smallest projection of positive feature vectors onto the normal and the largest negative projection. We refer to the hyperplane defined by $\mathbf{w}$ and b in equations 5 and 6 as the discriminative hyperplane approximation (DHA).

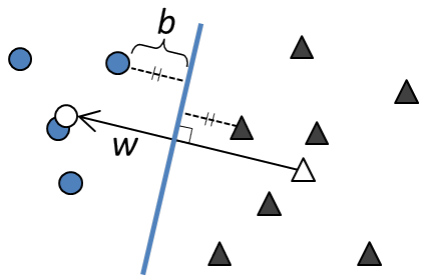

Figure 2. Blue circles represent positive feature vectors, black triangles represent negative feature vectors. The white circle is the mean positive vector and the white triangle is the mean negative vector. The vector pointing from the mean of positive vector to the mean negative feature vector usually gives a good normal for a separating hyperplane. The offset along the normal is then computed as the average of the minumum positive projection and the maximum negative projection.

In practice, to compute the normal $\mathbf{w}$, we found that L_2 normalizing the average positive feature vector (which were already individually L_2 normalized) but not normalizing the average negative feature vector (which were individually L_2 normalized) gave the best results.

This method can fail to find a hyperplane that separates the positive and negative feature vectors well, although it will always yield a valid hyperplane. Mainly, the failure mode occurs when the positive or negative data has a strong cluster of feature vectors plus some outliers. The strong cluster will be well separated, but the outliers may not be.

The DHA is very fast to compute and our experiments will show that it works well in practice.

4.2. Data Reduction

In this section we discuss other methods for speeding up computation that can be used in conjunction with any method for computing separating hyperplanes. We discuss two data reduction methods and combine them with hyperplane similarity to validate the effect on the calculation time and the accuracy.

The first method is to reduce the number of samples in the negative data. The k negative feature vectors closest to the mean positive data are extracted from the negative data by nearest neighbors. Since the SVM computation increases at least quadratically [1] with the number of samples, this method can greatly reduce the SVM calculation time. At the same time, until a certain number of nearest neighbors, this method only eliminates the feature vectors that do not support the separating hyperplane, and thus have no influence on the accuracy.

The second method is to reduce the dimension of the feature space. The principal components are extracted from the negative data and all feature vectors are projected onto

these principal components. The principal components and the dimension reduced negative data need only be calculated once for the verification test. Thereby, the only additional calculation for testing is the projection of the positive feature vectors onto the principal components, which is relatively fast.

5. Datasets and Evaluation Protocols

IARPA Janus Benchmark-A (IJB-A) IJB-A [9] is a face verification and identification dataset, containing images captured from unconstrained environments with wide variations of pose and imaging conditions. There are 500 identities with a total of 25,813 images (5,397 still images and 20,412 video frames sampled from 2,042 videos).

A set of images for a particular identity is called a template. Each template can be a mixture of still images and sampled video frames. The numbers of images (or frames) in a template ranges from 1 to 190 with approximately 10 images per template on average. There are 10 training and testing splits. Each split contains 333 training and 167 testing identities. The training data are not used to train the CNN, but used as the negative data for hyperplane estimation.

As a feature extractor for IJB-A images, we used the VGG-Face CNN architecture and parameters from Parkthi et al. [13]. This network produces feature vectors of length 4096. The dataset provides a ground truth bounding box for each face with 3 landmarks: center of both eyes and a nose base. For our results, we cropped the images with the bounding box enlarged by 1.1 from the provided size. No 2D face alignment is applied, since it had a negligible effect on the accuracy. Input images are first resized to 256x256 and cropped to 224x224 from the image center to match the CNN input size, and the mean pixel value of the VGG Face training dataset [13] is subtracted from each pixel.

YouTubeFaces YouTubeFaces[24] is a video-based face recognition dataset. It contains 3425 videos of 1595 people collected from YouTube, with an average of 2 videos per identity and 181.3 frames per video. It contains 10 folds of 500 video pairs. One fold is chosen as test data, while others are used as the negative data for hyperplane estimation.

As a feature extractor, we used the same VGG-Face model that we used for IJB-A. The dataset provides a ground truth bounding box for each face. We cropped the face with the given bounding box, and did not apply any 2D face alignment. Input images are resized to 256x256 and cropped to 224x224 from the image center, and the mean pixel value of the training data, VGG Face dataset [13], is subtracted from each pixel.

When testing, all the features from the same video sequence are averaged due to computational efficiency. We also tested sampling multiple frames from the video, and dividing video frames into subsets of frames and averaging the features from each subset, but both did not give any improvement to the result.

CUHK03 CUHK03 [10] is a pedestrian re-identification dataset that contains 14,097 cropped images of 1,467 identities. Each identity is observed by two camera views and has 4.8 images on average for each view. The dataset provides two kinds of bounding boxes, automatically detected and manually labeled. We evaluated our model with the manually labled bounding boxes. The dataset is randomly split into test data with 100 identities and train data with 1,367 identities. The experiment is repeated with 10 random splits. We tested with the multi-shot protocol, where multiple images from one camera view are used as one query.

The ResNet-50 architecture [7] is used as a feature extractor. We took the parameters pre-trained on ImageNet [4], and fine-tuned with CUHK03 and Market1501 [27], which is another re-id dataset. We trained on two similar datasets to overcome the limited amount of training data in CUHK03 alone. Images are resized to 224x224 to match the CNN input size, and the mean image of the ImageNet data is subtracted from the resized image.

6. Results

In this section, we first show the verification results on IJB-A, YTF, and CUHK03 using L_2 distance as well as hyperplane similarity (HS). Results from both linear SVM and the discriminative hyperplane approximation (DHA) are reported. We evaluate the relation between accuracy and calculation time among the different methods including the effect of the data reduction methods described in Section 4.2.

6.1. Hyperplane similarity

IJB-A Figure 3 shows the ROC curves (showing true acceptance rate (TAR) versus false acceptance rate (FAR)) on the IJB-A dataset for three different verification methods: L_2 distance, and hyperplane similarity using linear SVM and using DHA. Numbers from the figure are given in Table 2 along with calculation times. The table also shows results from some state-of-the-art methods.

The calculation time is the time spent to calculate the similarity scores of all the test template pairs. It does not include the feature extraction time, since all the methods share the same process. Experiments were processed on the system equipped with Intel Xeon CPU (E5-2650 v4/2.20GHz), 256GB memory, and Ubuntu 14.04.4. All the processes were run with a single core. The algorithms were implemented in Python 2.7.2, and the sklearn package was used for solving linear SVM.

Both hyperplane similarity methods, SVM and DHA, achieved better results compared with the L_2 distance. The

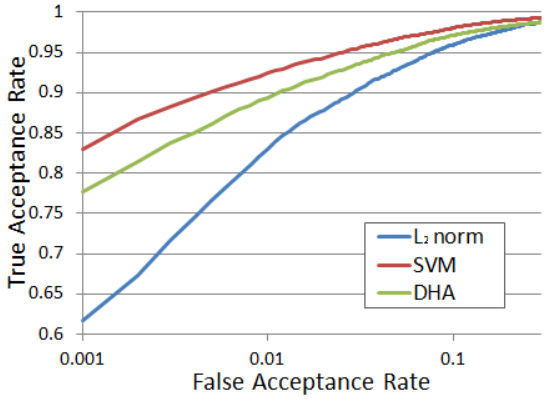

Figure 3. ROC curves for IJB-A dataset. Both hyperplane similarity methods achieve much better results than L_2 distance for lower false acceptance rates (which are the preferred operating regime).

improvement is greatest for lower FAR rates, which are more useful in practice. At FAR of 0.001, the true acceptance rate increases from 61.7% for L_2 distance to 83.0% (+21.3%) for hyperplane similarity using SVM or 77.7% (+16%) for hyperplane similarity using DHA. This is a substantial increase for a face verification system.

Although the result using DHA degrades somewhat compared with SVM, the computation time is significantly less (from 450.7 seconds for SVM to 27.8 seconds for DHA). Compared with the state-of-art results, hyperplane similarity achieves comparable results especially considering that the network was trained with regular softmax loss.

method	Verification (TAR@FAR)			time
	0.1	0.01	0.001	(sec)
L_2 distance	0.959	0.830	0.617	9.6
HS+SVM	0.980	0.924	0.830	450.7
HS+DHA	0.971	0.894	0.777	27.8
VGG-Face [13]	0.937	0.805	0.604	-
Crosswhite et al. [3]	-	0.939	-	-
TPE [15]	0.964	0.900	0.813	-
NAN [26]	0.978	0.941	0.881	-

Table 2. Results on IJB-A dataset. The hyperplane similarity methods have higher accuracy than L_2 distance. The result on DHA degrades compared with SVM, but the computation speed is significantly faster. The hyperplane similarity methods are competitive with the current state-of-the-art results shown in the bottom part of the table.

YouTubeFaces Figure 4 shows the ROC curve on the YTF dataset with three verification methods. In Table 3, TAR for each given FAR, equal error rate (EER), and the calculation time are reported.

Again, we see that hyperplane similarity is significantly more accurate than the L_2 distance for low false acceptance rates. For FAR equal to .001, the true acceptance rate increases by about 13 percentage points over the L_2 norm for hyperplane similarity with SVM and by about 10 percentage points with DHA. There is not much difference in EER among the three verification methods since the EER occurs at a relatively large FAR. Also, as with the results on IJB-A, the accuracy with DHA degrades compared with SVM but the computation time is significantly shorter.

Table 4 shows the results from state-of-art methods. VGG-Face [13] with no embedding learning is using the exact same CNN feature extractor and distance metric (L_2 distance) as we use in our results. We suppose the difference in our results is caused by a difference in the 2D face alignment and the frame selection within a video sequence. Unfortunately, other papers do not report TAR and FAR on this dataset.

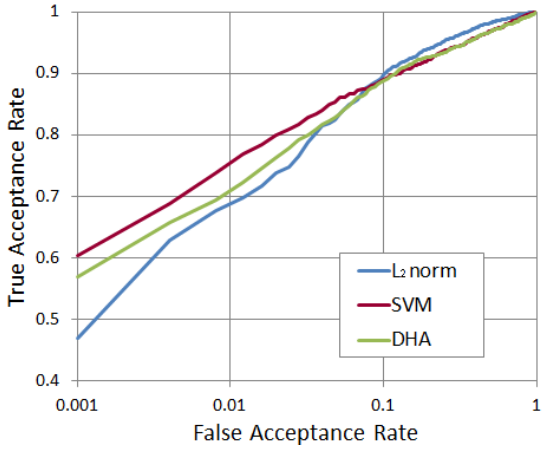

Figure 4. ROC curves for YTF dataset. Both hyperplane similarity methods achieve higher accuracy for lower FAR.

method	EER	TAR@FAR			time(sec)
		0.1	0.01	0.001	
L_2 distance	0.899	0.893	0.678	0.470	0.3
HS+SVM	0.892	0.888	0.738	0.604	89.6
HS+DHA	0.893	0.885	0.694	0.569	0.7

Table 3. Results on YTF dataset. Once again, hyperplane similarity is significantly more accurate than L_2 distance for low FAR. Using DHA greatly improves speed over SVM with a modest loss of accuracy.

CUHK03 Figure 5 shows the ROC curves on the CUHK03 dataset with three verification methods. In Table 5, the TAR for given FAR, rank-1 identification accuracy, and the calculation time are reported. Since CUHK03 is an identification problem, one probe identity is compared with 100 gallery identities. TAR is calculated by regarding them as 100 verification problems.

method	EER	ACC
VGG-Face* [13]	0.928	0.916
VGG-Face [13]	0.974	0.973
NAN [26]	-	0.957
FaceNet [16]	-	0.951
SphereFace [11]	-	0.950
L_2 distance	0.899	0.905
HS+SVM	0.892	0.908
HS+DHA	0.893	0.902

*Result without embedding learning.

Table 4. Comparison with state-of-art results on YTF dataset. TAR for low FAR, for which hyperplane similarity works well, cannot be directly compared since other methods do not report these numbers. EER and ACC measures emphasize higher FAR, for which hyperplane similarity does not improve over L_2 distance.

For each hyperplane similarity method, results with two different negative sets are reported. One is the gallery data of the test set. The other is a subset of the training data. From the training data, we only used images from the gallery camera view, and discarded images from the probe camera view.

The results on CUHK03 are similar to results on YTF, where both hyperplane similarity methods achieve better TAR accuracy with FAR lower than 0.1, but the improvement over L_2 distance is smaller than with YTF.

Using gallery data as the negative set yields better accuracy compared to using training data in this case. By using the gallery data, the hyperplane estimation can take advantage of the latent information in the distribution of the candidate identities. The usage of this approach (negative data = gallery) is limited to applications in which a fixed and reasonably large gallery set can be defined such as for face identification applications, but not typically for face verification applications.

Table 6 shows the results from the state-of-art methods. True acceptance rates for low false acceptance rates cannot be directly compared since they are not reported by most of the methods.

method	rank1	TAR@FAR			time
		0.1	0.01	0.001	(sec)
L_2 distance	0.743	0.963	0.782	0.460	0.9
HS+SVM (train)	0.758	0.968	0.821	0.501	137.5
HS+SVM (gal)	0.741	0.963	0.838	0.567	17.2
HS+DHA (train)	0.759	0.971	0.816	0.464	5.1
HS+DHA (gal)	0.741	0.977	0.843	0.476	2.7

Table 5. Results on CUHK03 dataset. The SVM method has higher accuracy for lower FAR, but the improvement over L_2 distance is smaller than with the other two datasets.

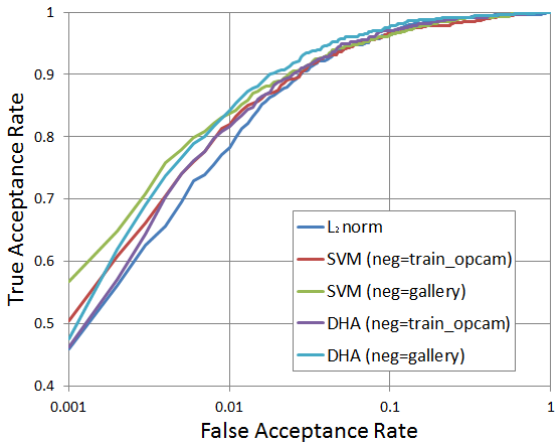

Figure 5. ROC curves on CUHK03 test set for L_2 distance and both hyperplane similarity methods. Once again, we see improvements over L_2 distance mainly for lower false acceptance rates. Improvements are greatest when the gallery set is used as the negative set for hyperplane estimation.

method	rank1	rank5	rank10
Zheng et al. [28]	0.883	0.957	0.978
Deep Transfer Learning [5]	0.854	-	-
LSRO [29]	0.846	0.976	0.989
CRAFT [2]	0.843	0.971	0.983
Domain Guided Dropout [25]	0.805	0.949	0.971
L_2 distance	0.743	0.963	0.973
HS+SVM (train)	0.758	0.968	0.966
HS+SVM (gallery)	0.741	0.963	0.959
HS+DHA (train)	0.759	0.971	0.972
HS+DHA (gallery)	0.741	0.977	0.971

Table 6. State-of-art results on CUHK03 dataset. Hyperplane similarity is mainly effective at improving TAR for low FAR and not at improving rank-N recognition rates. Unfortunately, results on CUHK03 typically only report rank-N recognition rates.

6.2. Data Reduction

To evaluate the relation between accuracy and calculation time, we experimented with two data reduction methods on the IJB-A dataset: PCA and nearest neighbors (NN). With PCA, results are reported with the feature dimension reduced to $d = 32$, 64, 128, 256, 512, and 1024. With NN, results are reported with the number of negative feature vectors reduced to 32, 64, 128, 256, 512, and 1024. For both data reduction methods, we tested using both hyperplane estimation methods: SVM and DHA. The calculation time does not include the feature extraction, the calculation of principal components, nor the time of dimension reduction of the negative data, since all the methods share the same process or need to be processed only once before the evaluation. Figure 6 is the plot of the TAR for FAR=0.001 and

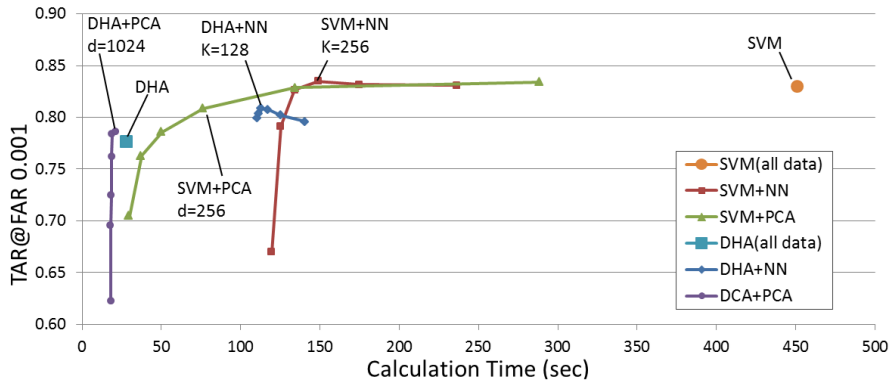

Figure 6. The verification accuracy and the calculation time on IJB-A with negative data reduction.

the calculation time for all the experimental results. Table 7 shows some of the results in detail.

SVM+NN with K=256 achieved the best accuracy of all experiments, 83.5%, and reduced 74.3% of the computation time compared with using all the data. When NN is performed with SVM, it does not reduce accuracy until a certain point, 256 in this case, since it only eliminates the negative data that do not support the hyperplane anyway.

The accuracy of SVM+PCA gradually decreases as the number of dimensions gets smaller, due to the lower explained variance. Most results are inbetween the SVM (all data) and DHA (all data) results considering both verification accuracy and the calculation time.

DHA+PCA with d=1024 gave a small benefit for both accuracy and calculation time. The combination of DHA+NN had a longer calculation time compared with using all the negative data, since the NN calculation time was larger than what was reduced by data reduction.

As a whole, the combination of the hyperplane calculation method and the data reduction method gives a range of options to trade-off between accuracy and calculation time. The suitable option should be selected depending on the requirements of the application and the system setup.

method	TAR@FAR			time
	0.1	0.01	0.001	(sec)
L_2 distance	0.959	0.830	0.617	9.6
HS+SVM	0.980	0.924	0.830	450.7
HS+SVM (NN K=256)	0.980	0.923	0.835	116.1
HS+SVM (PCA d=512)	0.979	0.924	0.829	134.7
HS+DHA	0.971	0.894	0.777	27.8
HS+DHA (NN K=128)	0.971	0.907	0.809	119.4
HS+DHA (PCA d=1024)	0.972	0.898	0.786	22.1

Table 7. Results on IJB-A dataset with data reduction.

6.3. Discussion

The IJB-A dataset showed the biggest improvement in accuracy using hyperplane similarity compared to YTF and CUHK03. The major difference between these datasets are the separability of two templates in the feature space.

An IJB-A template consists of images from multiple randomly-selected media. Therefore, two same-identity templates include a wide variety of imaging conditions and are more likely to have large overlap in the distributions of their feature vectors. The estimated hyperplanes for each template will be more similar and achieve high hyperplane similarity. On the other hand, YTF is a verification between two video sequences, each video containing frames that are very similar to each other and thus contain less variety. This implies that two videos of the same person are less likely to have overlapping feature vector distributions. As a result, two templates from the same person are likely to be more separable in feature space implying that the hyperplane similarity will be smaller. It is similar with CUHK03, in which the probe and gallery are always from different camera views. However, for all the datasets we experimented on, hyperplane similarity achieved higher accuracy for low FAR compared with L_2 distance. Very low FAR is preferred in most verification applications such as export control and building security.

7. Conclusion

We have shown that using hyperplane similarity is an effective method to improve the accuracy of a previously trained CNN for face verification or person re-id. We demonstrated a fast algorithm for computing separating hyperplanes that is much faster than solving a linear SVM problem without sacrificing much accuracy. We also explored the speed versus accuracy trade-off using two data reduction methods. These techniques allow researchers to improve the accuracy of existing networks while taking into account computational efficiency.

References

[1] L. Bottou and C. Lin. Support vector machine solvers. In *Large scale kernel machines 3, no. 1*, pages 301–320, 2007. 4

[2] Y. C. Chen, X. Zhu, W. S. Zheng, and J. H. Lai. Person re-identification by camera correlation aware feature augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2017. 7

[3] N. Crosswhite, J. Byrne, O. M. Parkhi, C. Stauffer, Q. Cao, and A. Zisserman. Template adaptation for face verification and identification. *CoRR*, abs/1603.03958, 2016. 2, 6

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 5

[5] M. Geng, Y. Wang, T. Xiang, and Y. Tian. Deep transfer learning for person re-identification. *CoRR*, abs/1611.05244, 2016. 7

[6] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning and invariant mapping. In *IEEE Transactions on Computing*, 2006. 2

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 5

[8] F. J. Huang and Y. LeCun. Large-scale learning with svm and convolutional nets for generic object categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006. 2

[9] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1931–1939, 2015. 1, 2, 5

[10] W. Li, R. Zhao, T. Xiao, and X. Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *CVPR*, 2014. 5

[11] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. SphereFace: Deep Hypersphere Embedding for Face Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 7

[12] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 507–516, 2016. 2

[13] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, volume 1, page 6, 2015. 1, 2, 5, 6, 7

[14] R. Ranjan, C. D. Castillo, and R. Chellappa. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*, 2017. 2

[15] S. Sankaranarayanan, A. Alavi, C. D. Castillo, and R. Chellappa. Triplet probabilistic embedding for face verification and clustering. In *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–8, Sept 2016. 6

[16] 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. 1, 7

[17] 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. 2

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[19] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*, 2014. 2

[20] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1891–1898, 2014. 1

[21] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 1, 2

[22] Y. Tang. Deep learning using linear support vector machines. In *Proceedings of the International Conference on Machine Learning*, 2013. 2

[23] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision (ECCV)*, pages 499–515, 2016. 1, 2

[24] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE, 2011. 5

[25] T. Xiao, H. Li, W. Ouyang, and X. Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1249–1258, 2016. 7

[26] J. Yang, P. Ren, D. Zhang, D. Chen, F. Wen, H. Li, and G. Hua. Neural aggregation network for video face recognition. In *Proceedings of the 32th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7

[27] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In *Computer Vision, IEEE International Conference on*, 2015. 5

[28] Z. Zheng, L. Zheng, and Y. Yang. A discriminatively learned CNN embedding for person re-identification. *CoRR*, abs/1611.05666, 2016. 7

[29] Z. Zheng, L. Zheng, and Y. Yang. Unlabeled samples generated by GAN improve the person re-identification baseline in vitro. *CoRR*, abs/1701.07717, 2017. 7