One Shot Similarity Metric Learning for Action Recognition

Orit Kliper-Gross[1], Tal Hassner[2], and Lior Wolf[3]

[1] The Department of Mathematic and Computer Science, The Weizmann Institute of Science, Rehovot, Israel.
orit.kliper@weizmann.ac.il
[2] The Department of Mathematics and Computer Science, The Open University, Raanana, Israel.
hassner@openu.ac.il
[3] The Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel.
wolf@cs.tau.ac.il

Abstract. *The One-Shot-Similarity (OSS) is a framework for classifier-based similarity functions. It is based on the use of background samples and was shown to excel in tasks ranging from face recognition to document analysis. However, we found that its performance depends on the ability to effectively learn the underlying classifiers, which in turn depends on the underlying metric.*
In this work we present a metric learning technique that is geared toward improved OSS performance. We test the proposed technique using the recently presented ASLAN action similarity labeling benchmark. Enhanced, state of the art performance is obtained, and the method compares favorably to leading similarity learning techniques.

Keywords: Learned metrics; One-Shot-Similarity; Action Similarity

1 Introduction

Analyzing videos of actions performed by humans is a subject of much research in Computer Vision and Pattern Recognition. The particular problem of action pair-matching is the task of determining if actors in two videos are performing the same action or not. This, when the two actors may be different people and when the viewing conditions may vary. Contrary to related image-similarity tasks such as pair-matching of face images [1], where class labels are well defined, this task is often ill-posed; actions are frequently not *atomic*, and so whether or not two videos present *same* or *not-same* actions is not well defined. In addition, when the videos are obtained "in the wild", with no control over viewing conditions and without the collaboration of the actors appearing in them, the task is even more challenging.

In this paper we focus on pair-matching (same/not-same classification) of action videos obtained in such unconstrained conditions. Performance in this task ultimately depends on the suitability of the similarity measure used to compare video pairs. Recent results on similar image-based challenges have shown

that employing *background information* (sometimes called *side information*) can boost performance significantly. In our framework, the background information consists of a moderately large set of unlabeled examples, that are expected to be of different classes than the pair of samples we are comparing.

Specifically, the One-Shot-Similarity (OSS) measure [2] utilizes unlabeled non-class examples to obtain better estimates for the similarity of two face images [3]. OSS results consequently outperformed other methods on the LFW challenge [4]. OSS also compares favorably to other metric learning techniques in tasks related to ancient document analysis [5] and elsewhere [2].

Here, we report attempts to employ OSS on the recently introduced "Action Similarity Labeling" (ASLAN) data set [6], which includes thousands of videos from the web, in over 400 complex action classes. The ASLAN set was designed to capture the variability typical to unconstrained, "in the wild", action recognition problems and is currently the most comprehensive benchmark available for action similarity in videos (some example frames from the ASLAN set are presented in Figure 1).

Our tests on the ASLAN benchmark demonstrate that the performance gain obtained using OSS and background information for other tasks does not carry over to action similarity on the ASLAN set. While background-information might capture information vital for correctly measuring the similarity of two actions, benefiting from this information requires that the input space is suitable of this type of analysis. We therefore propose a novel scheme for supervised metric learning, the OSS-Metric Learning (OSSML). OSSML learns a projection matrix which improves the OSS relation between the example same and not-same training pairs in a reduced subspace of the original feature space. Our results demonstrate that OSSML significantly enhances action similarity performance on the ASLAN benchmark, compared to existing state-of-the-art techniques.

To summarize, this work makes the following contributions: (a) We have developed a new metric learning approach and applied it to the problem of action similarity (pair-matching) in videos. (b) We show how learned projections using background statistics enhance the performance over unsupervised metric learning (such as PCA). (c) We further show that applying two complementary weakly supervised criteria in an interleaving manner provides a substantial boost in performance, obtaining state-of-the-art results on the ASLAN benchmark.

The rest of this paper is structured as follows. Section 2 presents the OSSML and derives its formulation. Section 3 applies OSSML to action recognition on the ASLAN benchmark. Experimental results are presented in section 4. We conclude in section 5.

1.1 Related work

Metric learning. The choice of a suitable metric is crucial for the design of a successful pattern recognition system. The literature on the subject is therefore substantial. Some existing similarity measures are hand crafted (e.g., [7, 8]). Alternatively, there is growing interest in methods which apply learning techniques to fit similarity measures and metrics to available training data (see [9]

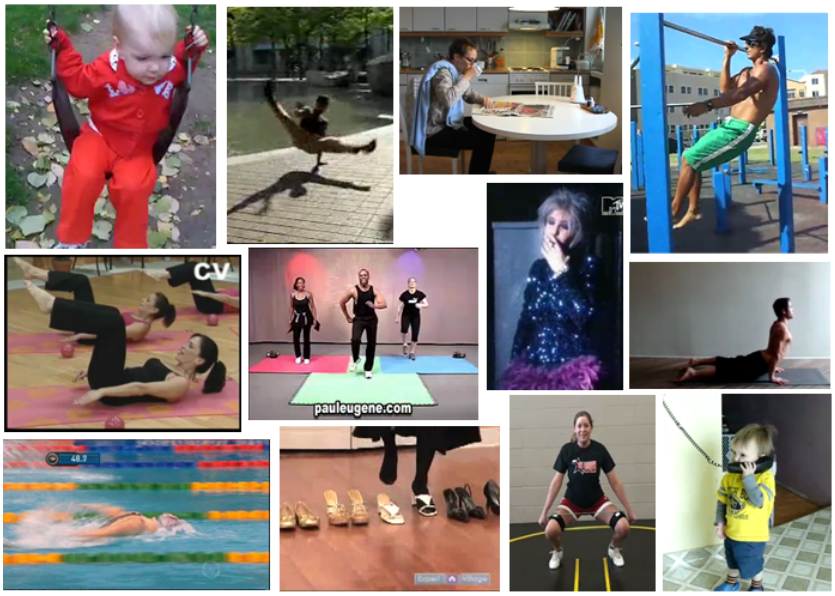

Fig. 1. Examples of actions in the ASLAN set

for a comprehensive study). Most common to these techniques is the learning of a projection matrix from the data so that the Euclidean distance can perform better in the new subspace. Learning such a matrix is equivalent to learning a Mahalanobis distance in the original space.

The Relevant Component Analysis (RCA) method of Bar-Hillel et al. [10] is one such example. They learn a full rank Mahalanobis metric by using equivalence constraints on the training elements. Goldberger et al. [11] described the Neighborhood Component Analysis (NCA) approach for k-NN classification. NCA works by learning a Mahalanobis distance minimizing the leave-one-out cross-validation error of the k-NN classifier on a training set. Another method, designed for clustering by [12], also learns a Mahalanobis distance metric, here using semi-definite programming. Their method attempts to minimize the sum of squared distances between examples of the same label, while preventing the distances between differently labeled examples from falling below a lower bound.

In [13] a Large Margin Nearest Neighbor (LMNN) method was proposed, which employed semi-definite learning to obtain a Mahalanobis distance metric for which any collection of k-nearest neighbors always has the same class label. Additionally, elements with different labels were separated by large margins.

The Information Theoretic Metric Learning (ITML) approach of Davis et al. [14] solves a Bregman's optimization problem [15] to learn a Mahalanobis distance function. The result is a fast algorithm, capable of regularization by a known prior matrix, and is applicable under different types of constraints, including similarity, dissimilarity and pair-wise constraints. The Online Algorithm for

Scalable Image Similarity (OASIS) [16] was proposed for online metric learning for sparse, high-dimensional elements.

Unlike the previously mentioned approaches, the recent method of Nguyen and Bai [17] attempts to learn a cosine similarity, rather than learning a metric for the Euclidean distance. This was shown to be particularly effective for pair-matching of face images on the Labeled Faces in the Wild (LFW) benchmark [1, 18].

Similarities employing background information. The first similarity measure in a recent line of work, designed to utilize background-information, is the One-Shot-Similarity (OSS) of [3, 2]. Given two vectors I and J, their OSS score is computed by considering a training set of background sample vectors N. This set of vectors contains examples of items different from both I and J, but are otherwise unlabeled. We review the OSS score in detail in Sec. 2.1. This OSS has been shown to be key in amplifying performance on the LFW data set. Here, we extend the OSS approach by deriving a metric learning scheme for emphasizing the separation between same and not-same vectors when compared using the OSS.

2 One-Shot-Similarity Metric Learning (OSSML)

Given a set of training examples our goal is to learn a transformation matrix which improves OSS performance, as measured using cross-validation. We next derive this transformation for the case where the classifier underlying the OSS computation is a free-scale Fisher Linear Discriminant.

2.1 The free-scale LDA-based, symmetric OSS score

Given two vectors I and J their One-Shot-Similarity (OSS) score is computed by considering a training set of background sample vectors N. This set contains examples of items not belonging to the same class as neither I nor J, but are otherwise unlabeled. A measure of the similarity of I and J is then obtained as follows: First, a discriminative model is learned with I as a single positive example, and N as a set of negative examples. This model is then used to classify the vector, J, and obtain a confidence score. A second such score is then obtained by repeating the same process with the roles of I and J switched. The particular nature of these scores depends on the classifier used. The final symmetric OSS score is the average of these two scores. Figure 2 summarizes these steps.

The OSS score can be fitted with almost any discriminative learning algorithm. In previous work, Fisher Linear Discriminant (FLD or LDA) [19, 20] was mostly used as the underlying classifier. Similarities based on LDA can be efficiently computed by exploiting the fact that the background set N, which is the source of the negative samples, is used repeatedly, and that the positive class, which contains just one element, does not contribute to the within class covariance matrix.

```
One-Shot-Similarity(I, J,  N) =

     Model1 = train(I, N)
     Score1 = classify(J, Model1)

     Model2 = train(J, N)
     Score2 = classify(I, Model2)

     return ½(Score1+Score2)
```

Fig. 2. Computing the symmetric One-Shot-Similarity score for two vectors, **I** and **J**, given a set, **N**, of background examples.

The free-scale LDA-based One-Shot-Similarity is a simplified version in which the projection is done along the unnormalized vector. Although in general, the OSS score is not a positive definite kernel, it was shown in [2] that the free-scale LDA-based OSS version gives rise to a positive definite kernel and so is suitable for use in kernel machines, such as Support Vector Machines (SVM) [21]. The symmetric Free-Scale One-Shot-Similarity (FSOSS) between two vectors I and J given the negative set N, is expressed as:

$$FSOSS(I,J,N) = (I-\mu_N)^T S_w^+ (J - \frac{I+\mu_N}{2}) + (J-\mu_N)^T S_w^+ (I - \frac{J+\mu_N}{2}) \quad (1)$$

Where,
μ_N is the mean of the negative set with $X_1, ...X_r$ samples, and S_w^+ is the pseudo-inverse of $S_w = \frac{1}{r}\sum_{k=1}^{r}(X_k-\mu_N)(X_k-\mu_N)^T$. In practice, to allow for efficient computation, we apply PCA before the learning, and therefore there are more examples than dimensions, thus, S_w is invertible and S_w^+ is simply the inverse $(S_w)^{-1}$, which we will denote by, $(S_w)^{-1} = S^{-1}$.

2.2 Deriving the OSSML

Let $I_i, J_i \in R^n$ be the pairs of input vectors in the training set. Let $L_i \in \{0,1\}$ be the corresponding binary labels indicating if I_i and J_i belong to the same class or not. Our goal is to learn a linear transformation $A : R^n \longrightarrow R^m (m < n)$ which will be used to compute OSS in the transformed space. Specifically, we want to learn the linear transformation that will minimize the cross-validation error when similarities are computed by the OSSML score below. For each pair of vectors I, J, the OSS score in the transformed space (i.e. OSSML) is defined by:

$$OSSML(I, J, N, A) = \\ (AI - \mu_{AN})^T S_{AN}^{+}(AJ - \frac{AI+\mu_{AN}}{2}) + (AJ - \mu_{AN})^T S_{AN}^{+}(AI - \frac{AJ+\mu_{AN}}{2}) \quad (2)$$

Here, N is the negative set, with r samples, A is the matrix to learn, AN is the negative set after applying A to each vector, μ_{AN} is the mean vector of the negative set after applying A, S_{AN}^{+} is the pseudo-inverse of $S_{AN} = \frac{1}{r}\sum_{k=1}^{r}(AX_k - \mu_{AN})(AX_k - \mu_{AN})^T$, and $S_{AN} = AS_wA^T = ASA^T$ is invertible iff $S(= S_w)$ is invertible.

Replacing, S_{AN}^{+} by $S_{AN}^{-1} = (ASA^T)^{-1}$ We get,

$$\begin{aligned} &OSSML(I, J, N, A) = \\ &\\ &\tfrac{1}{2}(AI - A\mu_N)^T(ASA^T)^{-1}(2AJ - AI - A\mu_N) + \\ &\tfrac{1}{2}(AJ - A\mu_N)^T(ASA^T)^{-1}(2AI - AJ - A\mu_N) = \\ &\\ &\tfrac{1}{2}(I - \mu_N)^T A^T(ASA^T)^{-1}A(2J - I - \mu_N) + \\ &\tfrac{1}{2}(J - \mu_N)^T A^T(ASA^T)^{-1}A(2I - J - \mu_N). \end{aligned} \quad (3)$$

Using the following notations:

$$\begin{aligned} a &= (I - \mu_N) \\ b &= (2J - I - \mu_N) \\ c &= (J - \mu_N) \\ d &= (2I - J - \mu_N) \end{aligned}$$

We have,

$$OSSML(I, J, N, A) = \frac{1}{2}a^T A^T(ASA^T)^{-1}Ab + \frac{1}{2}c^T A^T(ASA^T)^{-1}Ad. \quad (4)$$

2.3 Objective function

The objective function $f(A)$ is defined by:

$$f(A) = \sum_{i \in Pos} OSS(I_i, J_i, N, A) - \alpha \sum_{i \in Neg} OSS(I_i, J_i, N, A) - \beta||A - A_0||^2 \quad (5)$$

Where, Pos and Neg are the set of indices of the pairs belong to the same and not-same sets, respectively. Our goal is to maximize $f(A)$ with respect to A, given two parameters α and β, both non-negative. In practice we iterate on a range of β values, using cross-validation on part of the training data, as suggested by the CSML [17] algorithm. For A_0 we followed [17] and tried different $m \times n$ initial projections.

2.4 Free-scale LDA-based OSS gradient

The objective function $f(A)$ is differentiable with respect to A. The gradient is given by:

$$\frac{\partial(f(A))}{\partial(A)} = \sum_{i \in Pos} \frac{\partial(OSS(I_i, J_i, N, A))}{\partial(A)} - \alpha \sum_{i \in Neg} \frac{\partial(OSS(I_i, J_i, N, A))}{\partial(A)} - 2\beta(A - A_0). \quad (6)$$

Using the notations in Equation 4, the free-scale LDA-based OSS derivative is given by,

$$\frac{\partial(OSS(I_i,J_i,N,A))}{\partial(A)} = \frac{\partial(\frac{1}{2}{a_i}^T A^T (ASA^T)^{-1} A b_i)}{\partial(A)} + \frac{\partial(\frac{1}{2}{c_i}^T A^T (ASA^T)^{-1} A d_i)}{\partial(A)}. \quad (7)$$

This consists of two identical terms. Each can be written as:

$$\frac{1}{2} \frac{\partial(x^T A^T (ASA^T)^{-1} A y)}{\partial(A)}$$

Denote by W the $(m \times n)$-dimensional matrix of the result obtained by deriving this term. Let $D = ASA^T$, where, A is an $(m \times n)$-dimensional matrix, S is an $(n \times n)$-dimensional matrix and thus, D is an $(m \times m)$-dimensional matrix.

We want to find the derivative of the function, $g(D, A) = x^T A^T D^{-1} A y$, with respect to the matrix A.

D is a function of A, thus the chain rule can be written as:

$$[\frac{\partial g(D)}{\partial A}]_{ij} = \frac{\partial g(D)}{\partial A_{ij}} = \sum_{k=1}^{K} \sum_{l=1}^{L} \frac{\partial g(D)}{\partial D_{kl}} \frac{\partial D_{kl}}{\partial A_{ij}} = Tr[(\frac{\partial g(D)}{\partial D})^T \frac{\partial D}{\partial A_{ij}}]$$

Which is a matrix of the same dimensions as A (i.e. $m \times n$).

The total derivative W is therefore,

$$W_{ij} = [\frac{\partial(x^T A^T (ASA^T)^{-1} A y)}{\partial A}]_{ij} = Tr[(\frac{\partial g(D)}{\partial D})^T \frac{\partial D}{\partial A_{ij}}] + [\frac{\partial g(D,A)}{\partial A}]_{ij} = Tr[(\frac{\partial(x^T A^T D^{-1} A y)}{\partial D})^T \frac{\partial D}{\partial A_{ij}}] + [\frac{\partial(x^T A^T D^{-1} A y)}{\partial A}]_{ij} \quad (8)$$

where, $\frac{\partial g(D)}{\partial D}$ and $\frac{\partial D}{\partial A_{ij}}$ are $(m \times m)$-dimensional matrices. The last term, $\frac{\partial(x^T A^T D^{-1} A y)}{\partial A}$,

gives a matrix the same size as A and we take the ij entry.

1. From the following identity (see, for example, [22] for the various identities used throughout)

$$\frac{\partial(x^T X^{-1} y)}{\partial X} = -X^{-T} x y^T X^{-T},$$

we have

$$\frac{\partial(x^T A^T D^{-1} A y)}{\partial D} = -D^{-1} A x (Ay)^T D^{-1} = -(ASA^T)^{-1} A x (Ay)^T (ASA^T)^{-1}$$

where, $X = D = ASA^T$ is an $(m \times m)$-dimensional symmetric matrix, and we use Ax and Ay instead of x and y.

2. Using the identity:

$$\frac{\partial(X^T B X)}{\partial X_{ij}} = X^T B J^{ij} + J^{ji} B X$$

we therefore have,

$$\frac{\partial D}{\partial A_{ij}} = \frac{\partial ASA^T}{\partial A^T_{ji}} = ASJ^{ji} + J^{ij} SA^T$$

Where, $X = A^T$, $B = S$, and J is a 4-dimensional tensor with $J^{il}_{jk} = \delta_{jl}\delta_{ki}$. J^{ji} is a matrix of the same dimensions as A^T which are, $(n \times m)$, with 1 at the ji entry, and 0 otherwise. We thus get a $(m \times m)$-dimensional matrix.

3. From the identity:

$$\frac{\partial b^T X^T D X c}{\partial X} = D^T X b c^T + D X c b^T$$

we get,

$$\frac{\partial x^T A^T D^{-1} A y}{\partial A} = D^{-T} A x y^T + D^{-1} A y x^T = (ASA^T)^{-1} A x y^T + (ASA^T)^{-1} A y x^T$$

where, $X = A$, $D = D^{-1} = (ASA^T)^{-1}$, $b = x$ and $c = y$.

Finally, the total derivative in Equation 8 becomes:

$$
\begin{aligned}
&W_{ij} = [\tfrac{\partial(x^T A^T (ASA^T)^{-1} Ay)}{\partial A}]_{ij} = \\
&Tr[(\tfrac{\partial(x^T A^T D^{-1} Ay)}{\partial D})^T \tfrac{\partial(ASA^T)}{\partial A_{ij}}] + \tfrac{\partial(x^T A^T D^{-1} Ay)}{\partial A} = \\
&Tr[(-(ASA^T)^{-1} Ax(Ay)^T (ASA^T)^{-1})^T (ASJ^{ji} + J^{ij} SA^T)] + \\
&((ASA^T)^{-1} Axy^T + (ASA^T)^{-1} Ayx^T)_{ij}
\end{aligned}
\tag{9}
$$

Which gives a scalar for each entry ij.

The general formula for W is given by,

$$
\begin{aligned}
&W(x,y)_{kl} = \\
&Tr[(-(ASA^T)^{-1} Ax(Ay)^T (ASA^T)^{-1})^T (ASJ^{lk} + J^{kl} SA^T)] + \\
&((ASA^T)^{-1} Axy^T + (ASA^T)^{-1} Ayx^T)_{kl}
\end{aligned}
\tag{10}
$$

for $k \in 1, ..., n$, $l \in 1, ...m$.

We have two such $(m \times n)$-dimensional W matrices for each (I_i, J_i) pair.

To summarize, Equation 6 becomes,

$$
\begin{aligned}
&\tfrac{\partial(f(A))}{\partial A} = \\
&\tfrac{1}{2} \sum_{i \in Pos} (W(a_i, b_i) + W(c_i, d_i)) - \\
&\tfrac{1}{2} \alpha \sum_{i \in Neg} (W(a_i, b_i) + W(c_i, d_i)) - \\
&2\beta(A - A_0)
\end{aligned}
\tag{11}
$$

With W as above (Equation 10) for,

$$
\begin{aligned}
a_i &= (I_i - \mu_N) \\
b_i &= (2J_i - I_i - \mu_N) \\
c_i &= (J_i - \mu_N) \\
d_i &= (2I_i - J_i - \mu_N)
\end{aligned}
$$

3 Application to Action Recognition

In this section we apply OSSML to action similarity by measuring its performance on the ASLAN dataset.

3.1 ASLAN data set

The Action Similarity Labeling (ASLAN) collection is a new action recognition data set. This set includes thousands of videos collected from the web, in over 400 complex action classes. To standardize testing with this data, a "same/not-same" benchmark is provided, which addresses the action recognition problem as a non class-specific similarity problem instead of multi-class labeling. Specifically, the goal is to answer the following binary question – "does a pair of videos present the same action, or not?". This problem is sometimes referred to as the "unseen pair matching problem" (see for example [1]). Each video in the ASLAN collection is represented using each of the following state-of-the-art video descriptors: HOG, HOF and HNF [23]. Below, we use these descriptors, as made available by [6] without modification.

3.2 Same/not-same benchmark

To report performance on the ASLAN database, the experimenter is asked to report aggregate performance of a classifier on ten separate experiments in a leave-one-out cross-validation scheme. Each experiment involves predicting the same/not-same labels for the video pairs in one of the ten "splits". Each such split includes 300 pairs of same actions and 300 pairs of not-same actions. In each experiment, nine of the splits are used for training, with the tenth split used for testing. The final parameters of the classifier under each experiment should be set using only the training data for that experiment, resulting in ten separate classifiers (one for each test set). The ASLAN benchmark has been designed such that these ten splits are mutually exclusive in the action labels they contain; if videos of a certain action appear in one split, no videos of that same action will appear in any other split. These tests therefore measure performance on general action similarity rather than the recognition of particular action classes.

3.3 Experimental setup

We apply our experiments on each of the three descriptors available with the ASLAN data set. The dimension of each of the three descriptors is 5000. For each descriptor, we begin by applying PCA to get the vectors in a reduced n-dimensional space. We preform extensive tests with different PCA dimensions to choose a suitable subspace. We further reduce the dimension by applying OSSML as follows.

For each of the ten separate experiments we divide the nine training subsets such that one subset was used as a negative set, four subsets as validation samples and four subsets as training samples. We then use the training samples to find a matrix A that maximize $f(A)$ for a given α, β and initial matrix $A0$. Then, we use the validation samples to choose the next matrix A such that the accuracy on the validation sets is increased. We proceed iteratively until convergence.

For comparison we have implemented the Cosine Similarity Metric Learning (CSML) algorithm following the description in [17]. We have further used the CSML projection as an initial projection for our own OSSML algorithm.

Finally we have used a combination of similarity scores produced by different descriptors in the projected subspace to find optimal classifiers using linear SVM [21].

Results are reported by constructing an ROC curve and measuring both the area under curve (AUC) and the averaged accuracy ± standard errors for the ten splits.

4 Experimental Results

We first apply LDA-based OSS in the original 5000-dimensional descriptor space and compare it to the Cosine Similarity (CS). Table 1 reports the results of finding an optimal threshold on similarities calculated between vectors in the original descriptor space, as well as on the square root values of the descriptor entries (which makes sense for histograms [3]). Original vectors were L2 normalized before similarities were computed.

Table 1. Original classification performance (no learning): Accuracy±Standard Error and (AUC), averaged over the 10-folds.

		OSS	**FSOSS**	**CS**
HOG	original	$53.75 \pm .0.5(54.6)$	$51.90 \pm 0.4(51.5)$	$54.27 \pm 0.6(55.7)$
	sqrt	$53.20 \pm .0.7(53.7)$	$52.22 \pm .0.6(50.6)$	$53.47 \pm .0.6(54.2)$
HOF	original	$53.52 \pm .0.5(55.8)$	$52.63 \pm 0.4(53.3)$	$54.12 \pm 0.7(56.5)$
	sqrt	$54.80 \pm .0.6(56.0)$	$52.58 \pm 0.6(52.9)$	$53.83 \pm 0.7(56.0)$
HNF	original	$54.57 \pm 0.5(55.6)$	$52.60 \pm 0.4(52.4)$	$54.50 \pm 0.6(57.6)$
	sqrt	$54.27 \pm .0.6(54.9)$	$53.17 \pm 0.6(51.5)$	$53.93 \pm 0.73(55.8)$

To allow for efficient computation, we next use PCA to reduce the dimension of the original space. PCA was preformed using different training sets for each experiment. We next choose an $n \times m$ initial projection matrix A_0 for the learning algorithm (in our setting $n = 100$ and $m = 50$). We tried three different initial projections as suggested by [17]. We found that in our case best initial results were obtained by a simple $n \times m$ PCA projection. The initial PCA projection already improved the results over the original vector space. See the first block of Table 2.

We next perform three metric learning scenarios: CSML and OSSML with initial PCA projection, as well as OSSML with the matrix obtained by the CSML algorithm as the initial projection. We apply the projections obtained by each of these scenarios and calculated both CS and OSS scores in the projected subspace.

In the next three blocks of Table 2 we report the performances achieved by finding optimal thresholds for each of these scores. In the last column, we show the performances achieved by concatenating the scores of the three descriptors and finding an optimal classifier using linear SVM on a three-dimensional input vector. We further concatenate both scores from all three descriptors to form a six-dimensional vector given as an input to the linear SVM to get an optimal classifier. This is reported as CS+OSS on the third line of each algorithm.

Table 2. Classification performance on ASLAN: Accuracy±Standard Error and (AUC), averaged over the 10-folds. Please see text for more details.

		HOG	HOF	HNF	all descriptors
PCA init.	CS	$60.08 \pm 0.7(63.9)$	$57.07 \pm 0.7(60.1)$	$60.43 \pm 0.7(64.2)$	$61.10 \pm 0.7(65.2)$
	OSS	$59.83 \pm 0.7(63.1)$	$56.88 \pm 0.6(59.4)$	$59.80 \pm 0.7(63.0)$	$60.98 \pm 0.7(64.9)$
	CS+ OSS				$61.23 \pm 0.6(65.4)$
CSML	CS	$60.15 \pm 0.7(64.2)$	$58.62 \pm 1.0(61.8)$	$60.52 \pm 0.6(64.3)$	$62.90 \pm 0.8(67.4)$
	OSS	$60.00 \pm 0.9(63.8)$	$58.88 \pm 0.7(62.4)$	$59.98 \pm 0.7(63.3)$	$62.63 \pm 0.7(67.6)$
	CS+ OSS				$63.12 \pm 0.9(68.0)$
OSSML after PCA	CS	$60.22 \pm 0.7(64.1)$	$57.20 \pm 0.8(60.5)$	$60.10 \pm 0.7(64.3)$	$60.80 \pm 0.6(65.7)$
	OSS	$60.05 \pm 0.7(63.8)$	$58.05 \pm 0.8(60.7)$	$60.53 \pm 0.8(64.0)$	$62.32 \pm 0.8(66.7)$
	CS+ OSS				$62.52 \pm 0.8(66.6)$
OSSML after CSML	CS	$60.63 \pm 0.6(65.0)$	$59.53 \pm 0.9(63.6)$	$60.83 \pm 0.8(65.1)$	$63.17 \pm 0.8(68.0)$
	OSS	$60.00 \pm 0.8(64.3)$	$60.05 \pm 0.5(63.8)$	$60.75 \pm 0.8(64.1)$	$63.70 \pm 0.8(68.9)$
	CS+ OSS				$64.25 \pm 0.7(69.1)$

ROC curves of the results for View-2 of the ASLAN data set are presented in Figure 3. The results were obtained by repeating the classification process 10 times. Each time, we use nine sets for learning as specified in Section 3.3, and evaluate the results on the tenth set. ROC curve was constructed for all splits together (the outcome value for each pair is computed when this pair is a testing pair).

To gain further insight on our results, Figure 4 presents the most confident predictions made by our best scoring OSSML based method. The figure presents the most confident *correct* same and not-same predictions, and the most confident *incorrect* same and not-same predictions. Here, confidence was measured

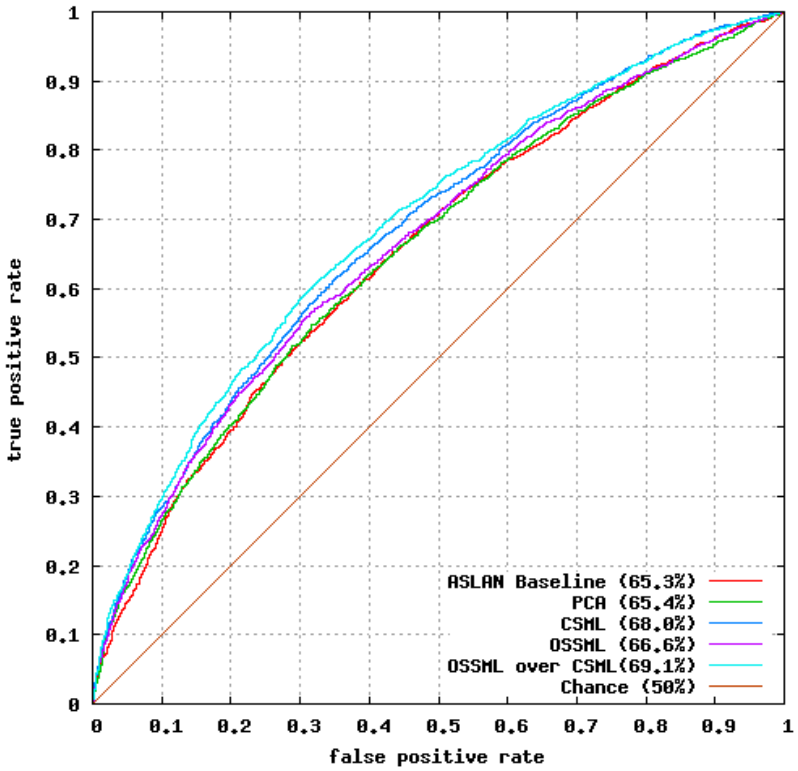

Fig. 3. ROC for the ASLAN benchmark

as the distance of the vector of similarities from the SVM hyperplane. These results emphasize the challenges of the ASLAN benchmark: as can be seen, many of the mistakes result from misleading context. Either "same" was predicted for two different actions because of similar background or camera motion, or "not-same" was predicted for the same action, based on very different backgrounds and motions.

5 Conclusion

In this paper we have extended the usability of the recently proposed One-Shot-Similarity to cases in which the underlying metric is such that this similarity is ineffective. To learn a new metric, we construct a cost function that encourages either high or low similarity to pairs of samples depending on the associated same/not-same label.

Experiments on a recent and challenging action recognition benchmark reveal that the proposed metric learning scheme is effective and leads to the best reported results on this benchmark; However, not surprisingly, the degree of success

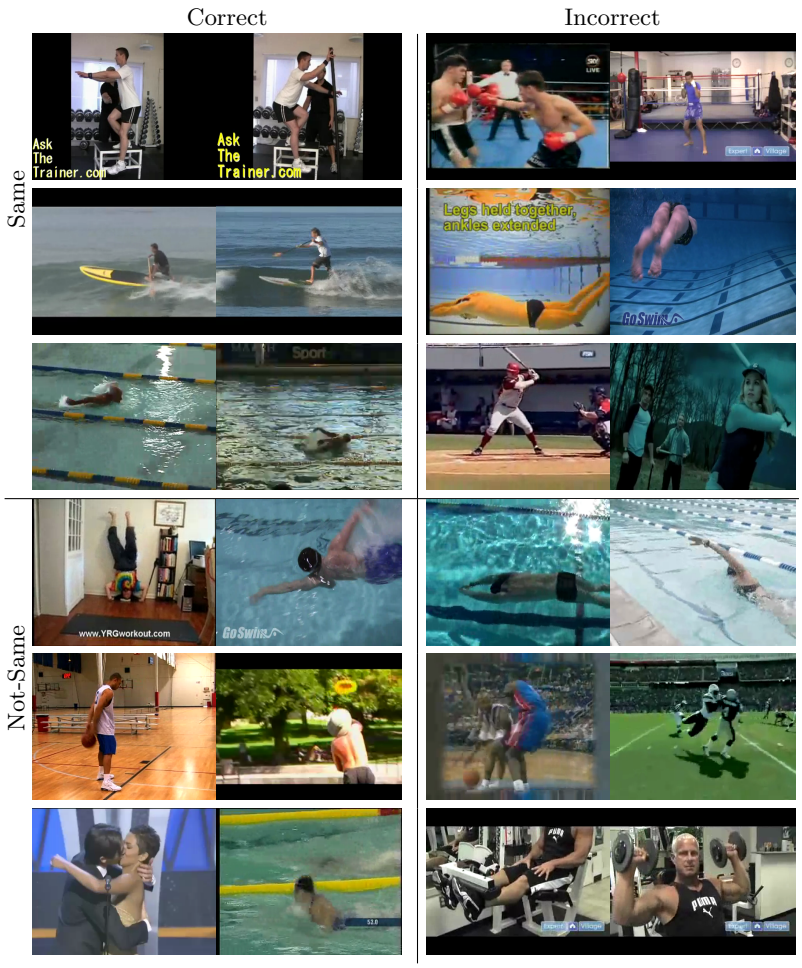

Fig. 4. Most confident OSSML results. The Same/Not-Same labels are the ground truth labels, and the Correct/Incorrect labels indicate whether the method predicted correctly. For example, the top right quadrant displays same-action pairs that were most confidently labeled as not-same.

depends on the specific initialization used. As an immediate but useful extension, we would like to apply similar methods to learn effective similarity scores between sets of vectors based on recent application of the One-Shot-Similarity to such problems [24].

References

1. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical

report, University of Massachusetts, Amherst, Technical Report 07-49 (2007)
2. Wolf, L., Hassner, T., Taigman, Y.: The one-shot similarity kernel. In: IEEE 12th International Conference on Computer Vision (ICCV). (2009) 897–902
3. Wolf, L., Hassner, T., Taigman, Y.: Descriptor based methods in the wild. In: Faces in Real-Life Images Workshop in European Conference on Computer Vision (ECCV). (2008)
4. Wolf, L., Hassner, T., Taigman, Y.: Similarity scores based on background samples. In: Asian Conference on Computer Vision (ACCV). (2009) 88–97
5. Wolf, L., Littman, R., Mayer, N., German, T., Dershowitz, N., Shweka, R., Choueka, Y.: Identifying join candidates in the Cairo Genizah. International Journal of Computer Vision (IJCV) (2011)
6. Kliper-Gross, O., Hassner, T., Wolf, L.: The action similarity labeling challenge. IEEE Transactions of Pattern Analysis and Machine Intelligence (TPAMI) (2011) Undergoing minor revisions.
7. Belongie, S., Malik, J., Puzicha, J.: Shape context: A new descriptor for shape matching and object recognition. In: Advances in Neural Information Processing Systems 13 (NIPS). (2001) 831–837
8. Zhang, H., Berg, A.C., Maire, M., Malik, J.: Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Volume 2. (2006) 2126–2136
9. Yang, L., Jin, R.: Distance metric learning: A comprehensive survey. (Michigan State Universiy) 1–51
10. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning distance functions using equivalence relations. In: International Conference on Machine Learning (ICML). Volume 20. (2003) 11–18
11. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighborhood components analysis. In: Advances in Neural Information Processing Systems 17 (NIPS). (2005) 513–520
12. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning with application to clustering with side-information. In: Advances in Neural Information Processing Systems 15 (NIPS). (2002) 505–512
13. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: Advances in Neural Information Processing Systems 18 (NIPS). (2006) 1473–1480
14. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: International Conference on Machine Learning (ICML). (2007) 209–216
15. Censor, Y., Zenios, S.A.: Parallel Optimization: Theory, Algorithms, and Applications. Oxford University Press (1997)
16. Chechik, G., Sharma, V., Shalit, U., Bengio, S.: Large scale online learning of image similarity through ranking. The Journal of Machine Learning Research (JMLR) **11** (2010) 1109–1135
17. Nguyen, H.V., Bai, L.: Cosine similarity metric learning for face verification. In: Asian Conference on Computer Vision (ACCV). (2010) 709–720
18. : (LFW results) `vis-www.cs.umass.edu/lfw/results.html`.
19. Fisher, R.: The use of multiple measurements in taxonomic problems. Annals of Human Genetics **7** (1936) 179–188
20. Hastie, T., Tibshirani, R., Friedman, J.H.: The elements of statistical learning. Springer (2001)

21. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20** (1995) 273–297
22. Petersen, K.B., Pedersen, M.S.: The matrix cookbook (2008) Version 20081110.
23. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2008) 1–8
24. Wolf, L., Hassner, T., Maoz, I.: Face recognition in unconstrained videos with matched background similarity. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). (2011)