A Time Pooled Track Kernel for Person Identification

Martin Bäuml Makarand Tapaswi Rainer Stiefelhagen
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
{baeuml, makarand.tapaswi, rainer.stiefelhagen}@kit.edu

Abstract

We present a novel method for comparing tracks by means of a time pooled track kernel. *In contrast to spatial or feature-space pooling, the track kernel pools base kernel results within tracks over time. It includes as special cases frame-wise classification on the one hand and the normalized sum kernel on the other hand. We also investigate non-Mercer instantiations of the track kernel and obtain good results despite its Gram matrices not being positive semidefinite. Second, the track kernel matrices in general require less memory than single frame kernels, allowing to process larger datasets without resorting to subsampling. Finally, the track kernel formulation allows for very fast testing compared to frame-wise classification which is important in settings where user feedback is obtained and quick iterations of re-training and re-testing are required. We apply our approach to the task of video-based person identification in large scale settings and obtain state-of-the art results.*

1. Introduction

Interest in automatic analysis of multimedia content has risen in recent years due to vastly increasing amounts of available data and the desire for automatic meta data generation, indexing and search to find relevant content.

An important aspect to consider is scale. Video data in the multimedia domain alone amounts to millions of hours of data, with hundreds of hours added per day. Efficiency, both in terms of computational and memory requirements, should therefore be taken into account. Recently, advances have been made for applying convex optimization methods for large scale problems [12]. For some classes of non-linear kernels, mapping to approximate feature maps $\hat{\Psi}(x)$ has been proposed in order to benefit from the speed-ups for linear SVMs [8, 15]. However, these techniques usually rely on an explicit feature map expansion which is impractical for very large or even infinite dimensional feature maps.

One important aspect of multimedia analysis is person identification [1, 3, 4, 13, 14] since it is not only relevant by itself, but also the basis for many higher level analysis tasks (*e.g.*, [10]). Person identification in video data is naturally performed on tracks, not individual frames, in order to average out noise and errors under the assumption that all frames of a track belong to the same individual.

The prevalent way is to regard a track as sequence of single frames. Thus, identification is usually performed by first classifying each frame separately and then reducing the individual results to a joint track classification estimate [4, 13, 14, 1]. Different classifiers for frame-level classification such as Nearest Neighbour [4], Multiple Kernel Learning [13] or Support Vector Machines [14] can be found in the literature. Similarly, different reduction schemes such as averaging [14, 1] or min-min[13] have been employed.

However, tracks can also be regarded as image sets. For image sets in general, specific set distances have been devised. We can differentiate between how approaches represent an image set, *e.g.* via its covariance matrix [17] or its convex hull [5], and the way image sets are compared, such as smallest distance between subspaces [5, 2] or correlation-based measures [18]. Kernels on sets are more prevalent in the context of local features, where an image or object is represented by a set of local features (*e.g.*, [16]). Robustness can for example be improved by non-uniform weighting [7]. The pooled NBNN kernel [9] pools base kernel comparisons on local features over sub-classes or visual-word-like clusters. This is closely related to our approach, however, it requires clustering in feature space which can be computationally expensive, whereas we do not.

Regarding tracks as image sets can be advantageous for multiple reasons: Image sets can be represented more compactly than the set of individual frames (*e.g.*, [5]). Especially for distance or Gram matrices, memory requirements reduce to the order of $O(M^2)$, where M is the number of *sets*, compared to $O(N^2)$ with N denoting the number of *features* across all sets combined (usually $M \ll N$). This is especially important in the context of large scale learning, where it might be unfeasible to keep all individual features (let alone a kernel's Gram matrix with memory requirement $O(N^2)$) in memory. Therefore, given a finite amount of memory, track representations can have more discriminative

power than (a subset of) individual frames. Furthermore, at test time, image set comparisons and decisions can be more efficient to compute, possibly at the expense of a one-time pre-computational step. This is desirable when quick iterations of training and testing are required, *e.g.* when user feedback is obtained and training cycles are performed with a human in the loop (see Sec. 3.2).

Our contributions are the following:

(1) We present a generalized time-based pooling kernel, which incorporates as special cases both the normalized sum kernel and frame-wise base kernels (Sec. 2). The kernel pools local kernel evaluations over time, leveraging the structure of tracks. This is very efficient in practice and can be implemented in a few lines of code.

(2) With the proposed time pooling kernel we are able to significantly speed up training-testing iterations (Sec. 3.2). Due to the structure of the kernel, classification/identification *of all tracks* can be performed simultaneously and efficiently by means of a single matrix multiplication. In particular, quick turn-around times allows efficient incorporation of feedback into the learning process.

(3) We present a large scale dataset for multimedia person identification, consisting of more than 21000 face tracks (1.2 million faces) (Sec. 3.1).

2. Time Pooled Kernels

We are interested in applying convex optimization methods to classification of time-based object sets (*e.g.*, face or person tracks). Many optimization methods refer to the data only within dot-products $\langle \mathbf{x}, \mathbf{y} \rangle$. A common way to augment features is to replace $\langle \mathbf{x}, \mathbf{y} \rangle$ by a kernel function $k(\mathbf{x}, \mathbf{y})$, which corresponds to computing the dot product in a different (usually higher dimensional) feature space: $k(\mathbf{x}, \mathbf{y}) = \langle \Psi(\mathbf{x}), \Psi(\mathbf{y}) \rangle$. Two examples of such kernelized classification schemes are the (two-class) Support Vector Machine with its convex objective

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{N} \sum_{i=1}^{N} \max\{0, 1 - y_i k(\mathbf{w}, \mathbf{x_i})\} \qquad (1)$$

and (multi-class) Multinomial Logistic Regression (MLR), which has the following convex objective:

$$\min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|^2 - \frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{M} \mathbf{1}[y_i{=}c] \ln \left(\frac{e^{k(\mathbf{w}_c, \mathbf{x}_i)}}{\sum_z e^{k(\mathbf{w}_z, \mathbf{x}_i)}} \right) \qquad (2)$$

where $\mathbf{x}_i$ and y_i denote features and corresponding class labels, N the number of training samples and M the number of classes.

Kernels are commonly defined on vectors since the relation to the dot product is evident. However, kernels can also be defined on other structures, *e.g.* strings or vector sets, thus making such structures available as input for above convex optimization schemes. We propose and evaluate a family of set kernels for *tracks*.

2.1. Pooling over time

Let $k(\mathbf{x}, \mathbf{y}) = \langle \Psi(\mathbf{x}), \Psi(\mathbf{y}) \rangle$ denote a *base* or *local kernel*, corresponding to a dot-product of individual frames $x \in X$ and $y \in Y$ in a (possibly infinite-dimensional) mapped feature space by the feature mapping Ψ.

Instead of using individual local kernel evaluations, we pool them over time. Let X and Y be time-consecutive sets of features, *e.g.* features extracted from tracks. We define a track kernel function as

$$K(X, Y) = K(X, Y; k(\cdot, \cdot), \Phi(\cdot)) \qquad (3)$$

which is built from two base functions, the local kernel for individual frames and a pooling function $\Phi(\cdot)$. This track kernel is a Mercer kernel (*i.e.*, positive semi-definite, p.s.d.) when the local kernel $k(\cdot, \cdot)$ is a Mercer kernel itself, *and* the pooling operation is from a set of operations (*e.g.*, sum) that preserve positive semi-definiteness [6].

For example, a local RBF kernel $k_{RBF}(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2})$ and $\sum\sum$ as pooling operation results in

$$K(X, Y) = \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2} \right), \qquad (4)$$

which is the normalized sum kernel [7] with a RBF kernel as base kernel. Since $k_{RBF}(\mathbf{x}, \mathbf{y})$ is a Mercer kernel, and $\sum$ is a valid construction operation, Eq. 4 is a Mercer kernel.

In contrast to [7, 9], the construction of feature sets by time does not require any a-priori clustering or nearest-neighbour search in feature space and thus is very efficient. This is based on the assumption that a) all features of one track belong to the same class/person, and b) the variation within a track is small.

The second assumption is obviously not always true, since for example in the case of faces, a person can turn his head within a track which usually amounts to changes in the extracted features. This is a drawback of the normalized sum kernel, as it possibly averages out few positive correspondences between the features of two tracks by many other negative correspondences.

We approach this problem from two perspectives. First, we employ non-averaging pooling operations $\Phi(\cdot)$ (*c.f.* Eq. 3), *e.g.* max. Second, we reduce these variations by applying the kernel not on full tracks, but on sub-tracks and thus improve the coherence of the features. Our goal is similar to [9], however, the pooling is performed over time. The shorter the pooling period is, the more coherent the sets are. In the extreme case this can go down to single frames, which we will discuss in Sec. 2.2.

Fixed-time splitting A first way of defining the pooling length is by splitting tracks into equal-length time-continuous sets of features. Let X be the original set of features for one track, then $X_i^{fixed(l)}$ are the new subtracks

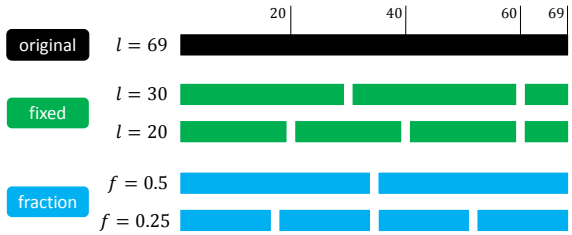


Figure 1: Different examples for splitting a track into fixed-time and fraction-time subtracks.

of equal length l (the last subtrack can be shorter when the original track length $len(X)$ is not a multiple of l). In most cases fixed-time splitting avoids a bias by track length.

Fraction-time splitting One possible issue with the fixed-time splitting is that long tracks get over-represented in the new subtrack sets. Instead of splitting into fixed-length sets, we can split each track into equal-size fractions $f \in (0, 1]$, *i.e.* construct $1/f$ subtracks $X_i^{frac(f)}$ from one track, each with $len(X_i^{frac(f)}) = f \cdot len(X)$. With fraction-time splitting, the relative number of tracks of each class is preserved. A visualization of the two different track splitting variants can be found in Fig. 1.

2.2. Special cases and relation to other methods

Normalized Sum Kernel and extensions As already discussed above, the normalized sum kernel is included as a special case (*c.f.* Eq. 4) with pooling operation $\sum\sum$ and no further splitting of tracks ($f = 1$). With the power-kernel $k(\cdot,\cdot)^p$ we obtain the soft-max Mercer kernel of [7].

Single-frame classification On the other extreme, by splitting all tracks into individual frames (*i.e.*, $l = 1$), we obtain the frame-wise classification as used, *e.g.*, in [1, 13, 14]. When employing $\sum\sum$ as pooling operation, track kernel and frame-wise classification are still similar, even for $l > 1$. In the frame-wise case (*e.g.*, [1]), classification results are averaged over the frames of the test-track (at test-time):

$$c^* = \arg\max_c \frac{1}{|T|} \sum_i^{|T|} \sum_m^W w_m^{(c)} k(\mathbf{x}_i, \mathbf{x}_m) \qquad (5)$$

where $|T|$ is the track length, W the number of kernel bases (individual features) from the training data, and $w_m^{(c)}$ the model parameters learned by minimization of Eq. 2. We neglect here for clarity, that in [1] the training features are subsampled and a sigmoid normalization step is performed before averaging over time.

For the track kernel-based classification (with $\sum\sum$ pooling) we compute for track T at test time

$$c^* = \arg\max_c \sum_m^W w_m^{(c)} K(T, T_m) \quad . \qquad (6)$$

Expanding $K(\cdot,\cdot)$ to the summation over local kernels $k(\cdot,\cdot)$, we obtain

$$c^* = \arg\max_c \sum_m^W w_m^{(c)} \frac{1}{|T||T_m|} \sum_i^{|T|} \sum_j^{|T_m|} k(\mathbf{x}_i, \mathbf{x}_j^m) \quad . \qquad (7)$$

By reordering the sums,

$$c^* = \arg\max_c \frac{1}{|T|} \sum_i^{|T|} \sum_m^W \frac{w_m^{(c)}}{|T_m|} \sum_j^{|T_m|} k(\mathbf{x}_i, \mathbf{x}_j^m) \qquad (8)$$

we can see, that due to the averaging over time, our formulation is the same for $l = 1$ (each pooling T_m set only contains a single frame, *i.e.* $|T_m| = 1$ and $\bigcup T_m$ corresponds to the individual frame kernel bases in Eq. 5). For $l > 1$, the difference is that all frames in a pooling set T_m share the parameter $w_m^{(c)}$, whereas in Eq. 5 we have one $w_m^{(c)}$ for each frame of the kernel basis.

Training/testing speed-up There is another subtle difference to single-frame classification. Since pooling over time is performed within the kernel, we can pre-compute significant parts of Eq. 8 required for both training (*c.f.* Eq. 2) *and* testing: Reordering the summation of Eq. 8 further and grouping all terms independent of $w_m^{(c)}$, we obtain

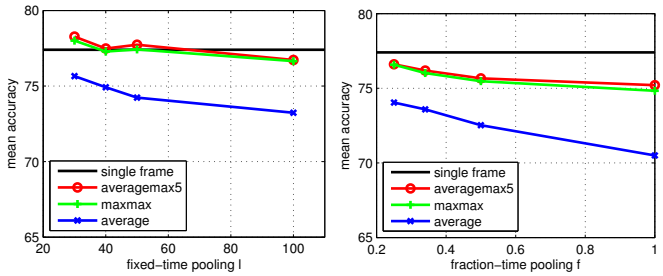
$$c^* = \sum_m^W w_m^{(c)} \frac{1}{|T||T_m|} \sum_i^{|T|} \sum_j^{|T_m|} k(\mathbf{x}_i, \mathbf{x}_j^m) \quad . \qquad (9)$$

The summation term over T and T_m can be pre-computed and stored once for all track combinations (this is the kernel's Gram matrix). We therefore avoid recomputing summations over individual frames at test time (compared to the single frame case, see Eq. 5). This results in a significant speed-up (by a factor of 5-10 in our experiments). A similar pre-computation could also be done for the single-frame case, but the much larger number of instances (frames instead of tracks) prohibits this for all practical instances.

2.3. Time and Space Complexity

Training The computation of the kernel's Gram matrix requires of $O(N^2)$ local kernel evaluations, where N is the number of features. Therefore, the computation of the full track kernel Gram matrix is in $O(N^2 \cdot k(d))$, with $k(d)$ being the complexity of the local kernel evaluation depending on feature dimensionality d. The memory requirements for the full Gram matrix depend on the pooling factors f or l, reducing the required memory compared to the frame-wise kernel by l^2, resulting in $O(\frac{N^2}{l^2})$. This allows, as argued before, to pre-compute and store the Gram's matrix for multiple rounds of training (*c.f.* Sec. 3.2).

Testing At test time, we benefit from the fact that pooling over each (sub-)track was already performed at training



(a) (b)

Figure 2: Influence of the pooling parameters. Comparison of mean accuracy on BBT 1-6 with different pooling functions for (a) fixed-time pooling and (b) fraction-time pooling. The black line denotes the single-frame recognition performance of [1].

time. For the case of MLR, this reduces classification to a single matrix multiplication $\mathbf{Kw}$ and obtaining the maximum over rows:

$$c_i^* = \arg\max_c \frac{e^{[\mathbf{K}]_i \mathbf{w}^{(c)}}}{\sum_z e^{[\mathbf{K}]_i \mathbf{w}^{(z)}}} = \arg\max_c [\mathbf{Kw}]_i \quad , \quad (10)$$

where $\mathbf{K} \in \mathbb{R}^{N/l} \times \mathbb{R}^{N/l}$ is the Gram matrix of the track kernel, $[\mathbf{K}]_i$ the ith row of $\mathbf{K}$, $|C|$ the number of classes, and $w \in R^{N/l} \times R^{|C|}$ the parameter vector obtained by minimizing the MLR loss function (Eq. 2). Thus, testing is dominated by the matrix multiplication $\mathbf{Kw}$ and results in time complexity of $O(\frac{N^2|C|}{l^2})$.

3. Evaluation

We evaluate the proposed kernels on the task of person identification in TV series. In order to compare results with previous methods, we use the KIT TV data set from [1], which consists of 6 episodes each of "The Big Bang Theory" (BBT) and "Buffy the Vampire Slayer" (BUFFY). The task is to assign identities to all face tracks. We employ the provided facial features and automatic speaker assignments, which are used as (noisy) labels for training the classifiers. 23.1% (BBT) and 20.3% (BUFFY) of all tracks are assigned a speaker identity (recall), with precision of 88% and 86.8%. There are 986 (BBT) and 1290 (BUFFY) labeled tracks, and 92495 and 105674 labeled frames. A full kernel matrix on all labeled frames has a memory requirement of $O(N^2)$ and thus would require $510GB$ and $665GB$ for BBT and BUFFY, respectively (for just 6 episodes of a single season). Results on the KIT TV data set can be found in Tbl. 1 and will be discussed in the following.

Baseline We compare against the SVM and MLR-based approaches from [1, 14] as reported in [1] on this data set. In [1], additional information such as unlabeled training data, constraints and/or information on clothing have been taken into account. Although such additional information could be considered for track-kernel-based classification as well, we would like to keep the evaluation as simple as possible and reduce the influence of unrelated components. We further compare against Covariance Discriminative Learning (CDL) [17], which is an example of track-based modeling and recognition. For CDL, tracks are represented by their covariance matrix. We follow [17] and add a small positive diagonal to the covariance matrix to ensure it is positive definite: $\mathbf{C}_X^* = cov(\mathbf{X}) + 10^{-3}\mathbf{I}$. The kernel between two tracks is defined as $k(\mathbf{X}, \mathbf{Y}) = \|\log(\mathbf{C}_X^*) \cdot \log(\mathbf{C}_Y^*)\|_F$. Using this kernel, we learn a MLR classifier (Eq. 2). CDL performs notably worse than the frame-based approaches with 68.2% on BBT and 59% on BUFFY vs. 77.4% and 65.8%, respectively.

Our approach We run our approach in different configurations. For a fair comparison, we use the same base kernel as in the baseline, a polynomial kernel of degree 2: $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^2$ (*c.f.* [1, 14]). We evaluate pooling over the full tracks (*i.e.*, $f = 1$), fixed-time pooling with $l = 30$ (split each track into equal-length sub-tracks of size 30, which corresponds to roughly one half of the average track length) and fraction-pooling with $f = 0.5$ (split each track into exactly 2 sub-tracks). We also compare different basic pooling strategies, namely *normalized sum* ($\frac{1}{MN}\sum\sum k(\cdot,\cdot)$), *single maximum* ($\max\max k(\cdot,\cdot)$), and *average N-max*, an average of the maximum N base kernel values ($\frac{1}{N}\sum \max_{1..N} k(\cdot,\cdot)$). We set $N = 5$ in our experiments. For an overview of the results see Tbl. 1.

Influence of splitting/time pooling For all variants of the track kernel, splitting tracks into sub-sets increases performance. A fixed-time splitting with $l = 30$ yields higher performance than fraction-time splitting with $f = 0.5$. One reason for this could be that with $f = 0.5$ even short tracks are split, whereas with $l = 30$ a track with less than 30 frames is not split. Interestingly, the non-Mercer kernel variants (due to the $\max$ operation) perform much better than the normalized sum-based kernel. One explanation might be that feature pairs with high local kernel scores are outnumbered by low-scoring pairs, for example due to pose mismatches or alignment problems.

Fig. 2 displays the influence of the two pooling parameters f and l for different pooling operations. We can see, that for both fixed-time and fraction-time splitting, a smaller subset size leads to higher recognition performance. This is not surprising since smaller subsets are more coherent, and thus learning can select from more representative subsets. Both max-based pooling operations perform similarly. The normalized sum kernel (denoted as *average* in the plot legend) on the other hand has a significant negative offset, possibly owing to the imbalance of good and bad matching feature pairs for a given track pair.

			BBT							BUFFY						
			1	2	3	4	5	6	**Avg.**	1	2	3	4	5	6	**Avg.**
single-frame SVM [14]			87.46	84.96	74.06	74.87	70.25	66.46	76.34	69.90	59.71	66.23	66.47	68.07	61.44	65.30
single-frame MLR [1]			88.59	87.61	76.18	74.01	72.76	65.24	77.40	68.85	61.37	65.96	67.19	69.85	61.72	65.82
track-repr CDL [17]			79.42	75.58	68.35	65.58	63.98	56.34	68.21	64.79	54.93	57.63	59.52	63.49	53.78	59.02
pool	split	Mercer														
$\frac{1}{MN}\sum\sum$	$f=1$	✓	82.15	77.88	70.47	67.99	65.59	58.90	70.50	62.04	55.45	56.15	62.75	63.49	56.18	59.34
$\frac{1}{MN}\sum\sum$	$f=0.5$	✓	84.57	80.53	72.43	69.88	67.74	60.00	72.52	63.74	57.84	59.48	63.59	64.12	57.56	61.06
$\frac{1}{MN}\sum\sum$	$l=30$	✓	86.66	84.42	74.06	74.01	71.86	62.93	75.66	68.32	58.46	62.53	66.11	69.21	59.32	63.99
$\max\max$	$f=1$		87.78	83.19	73.74	73.49	66.67	64.15	74.83	71.60	61.06	64.29	71.50	71.37	61.16	66.83
$\max\max$	$f=0.5$		89.07	84.07	73.74	74.18	67.74	64.02	75.47	71.20	62.82	66.42	70.78	69.72	61.35	67.05
$\max\max$	$l=30$		89.87	86.55	76.18	76.42	72.94	66.10	*78.01*	72.77	61.27	67.53	71.86	72.52	63.56	**68.25**
$\frac{1}{N}\sum\max_{1..N}$	$f=1$		88.59	83.36	74.23	74.01	67.03	64.02	75.21	71.20	61.06	64.20	72.10	71.76	60.98	66.88
$\frac{1}{N}\sum\max_{1..N}$	$f=0.5$		89.71	83.89	73.57	74.53	68.28	64.02	75.67	71.07	62.62	66.05	71.02	70.36	61.62	67.12
$\frac{1}{N}\sum\max_{1..N}$	$l=30$		90.35	86.37	76.35	77.11	73.12	66.34	**78.27**	72.77	60.44	67.62	70.78	72.52	63.10	*67.87*

Table 1: Results on the KIT TV data set [1]. Baseline results with single-frame methods are reported in the first two rows. CDL [17] (third row) is an example of a track-based representation method. The bottom section shows the performance of different instantiations of the time pooled track kernel. The $\max\max$ and average-N-max (with $N=5$) variants with a fixed-time splitting of $l=30$ perform best on average, despite not being Mercer kernels. The normalized sum variants (rows 1-3 bottom section) perform worst among the different track kernel variants, however, better than CDL. For all variants, splitting tracks into sub-sets increases performance, where a fixed-time splitting with $l=30$ consistently outperforms fraction-time splitting with $f=0.5$. We highlight best and second-best average performance.

EP	max-pr	SVM [14]	MLR [1]	avg-max $f=0.5$
1	30.26	72.69	69.36	74.10
2	19.68	63.89	61.97	63.29
3	19.13	66.06	64.07	66.93
4	25.49	65.37	67.09	67.20
5	34.86	66.22	63.21	71.05
6	15.59	65.19	61.83	63.46
7	20.31	53.01	50.30	53.01
8	21.89	70.27	68.78	71.52
9	19.15	60.40	56.87	59.58
10	29.49	68.97	65.77	69.36
11	14.51	59.97	59.09	58.92
12	20.13	52.93	52.38	52.22
13	19.19	70.25	66.50	69.44
14	27.91	54.77	52.79	59.53
15	20.96	57.83	58.59	58.50
16	19.97	65.52	62.85	64.50
17	14.17	68.77	67.72	67.85
18	18.33	59.36	57.13	61.59
19	17.20	67.74	68.49	68.17
20	19.07	56.77	55.10	56.59
21	12.20	59.96	56.91	60.47
22	19.84	60.23	58.27	59.20
Avg	20.88	63.01	61.14	**63.48**

Table 2: Accuracies for season 5 of BUFFY. The max-prior character is "Buffy" (first col.). With the *average* $\max_{1..N}$ track kernel and fraction-time pooling we outperform single-frame methods.

3.1. Large scale person identification

In order to test the scaling capabilities of the pooling kernel, we extended the KIT TV data set for BUFFY to span the full season[1]. Following [1], we performed face tracking, face alignment and block-based DCT feature extraction. Using fan transcripts, we perform automatic speaker assignment, and are able to assign speaker identities to 20.42% of the tracks of BUFFY with a precision of 77.47%.

Again, we compare single-frame approaches (MLR and SVM) against the time pooled track kernel (average $\max_{1..N}$ with $N=5$ and $f=0.5$). Results can be found in Table 2. We outperform both SVM and MLR with 63.48% compared to the next best 63% (SVM). Also, training and inference are very fast once the track kernel's Gram matrix has been computed. With our unoptimized MATLAB implementation, it takes only about $60s$ for both training *and* inference for the full season of BUFFY ($\sim$21000 tracks).

3.2. Learning with Feedback

An aspect that usually only plays a secondary role (in research) is the question of how to obtain 100% accurate labels from a given set of partly incorrect recognition results. The naïve solution would be to let a human manually correct all wrongly identified tracks. However, corrected results can be seen as new training data which allows us to train a better classifier. Re-training the classifier and re-testing all tracks after a few corrected samples can be beneficial. Due to the new training data, more originally wrongly classified tracks can now be identified correctly, which reduces the total amount of data that needs to be manually corrected. Note that this is different to the problem of *active learning*, which deals with the problem of which samples to select from a set of unlabeled samples [11].

We are interested in minimizing the time t_{total} to reach 100% accuracy. Important here is the relationship between training/inference *time* and the *number* of samples to correct each round. Our simple model of required labeling time is as follows. Let t_{pre} be a fixed amount of time needed to set up a classifier and pre-compute for example a kernel matrix. Let t_{init} be the time it takes to label one training sample before any learning has been performed, t_{fb} the feedback time on one wrongly classified sample, and t_{train} and t_{test} the time it takes to re-train the classifier and re-test all test samples, respectively. Further, let N be a number of labeled

[1] http://cvhci.anthropomatik.kit.edu/projects/mma

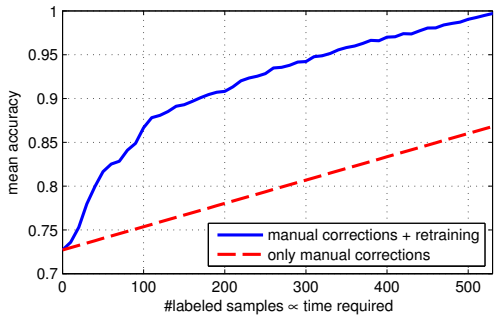


Figure 3: Training with feedback on BBT 1-6. We retrain/test every 10 corrected samples. After correcting roughly 500 samples, 100% accuracy is reached. Without re-training/testing, over 1000 samples would need to be manually corrected.

samples in each round. The total labeling time can then be computed as follows

$$t_{total} = t_{pre} + N_{init} \cdot t_{init} + k \cdot \max\{t_{train} + t_{test}, N \cdot t_{fb}\} \quad (11)$$

To minimize the number of unnecessarily manually corrected tracks, we should label $\hat{N}$ tracks in each round, such that $\hat{N} \cdot t_{fb} \approx t_{train} + t_{test}$, *i.e.* we use exactly the time for manual labeling while one train-test round is running. As discussed in Sec. 2.2, we can achieve fast training and inference with the track kernel, since we move the most time consuming step of kernel computation to t_{pre}. Therefore, $\hat{N}$ can be very small which reduces the amount of unnecessary labels.

In Fig. 3, we compare the time required on KIT TV BBT 1-6 to achieve 100% accuracy through feedback with and without repeatedly retraining the classifier after N labeled samples. Retraining *and* inference takes between 5 and 10 seconds on all 3759 tracks (compared to minutes for the single-frame classifier). In our experience, it is possible to correct on average 1 wrongly classified track per second with an appropriate user interface (*i.e.*, displaying many tracks at once). We therefore set $N = 10$. There were 1026 incorrectly classified tracks after the initial classifier run. We can see in Fig. 3 that re-training and inferring every 10 corrected samples, we reduce the amount of samples to correct to about 500, saving about half the time otherwise required to fully correct the initial recognition result.

4. Conclusion

We present a family of kernels to learn kernel-based classifiers on tracks and two strategies to efficiently pool base kernel results over time. We also investigate non-Mercer variants of the track kernel. The track kernel allows for fast training and inference compared to frame-wise classification, allowing for efficient integration of user feedback. We apply our approach to the task of large scale video-based person identification and obtain state-of-the art results.

Acknowledgments This work was supported by the German Federal Ministry of Education and Research (BMBF) under grant no. 13N12063.

References

[1] M. Bäuml, M. Tapaswi, and R. Stiefelhagen. Semi-supervised Learning with Constraints for Person Identification in Multimedia Data. In *CVPR*, 2013.

[2] H. Cevikalp and B. Triggs. Face recognition based on image sets. In *CVPR*, 2010.

[3] T. Cour, B. Sapp, and B. Taskar. Learning from Partial Labels. *JMLR*, 12(5):1225–1261, 2011.

[4] M. Everingham, J. Sivic, and A. Zisserman. "Hello! My name is... Buffy" Automatic naming of characters in TV video. In *BMVC*, 2006.

[5] Y. Hu, A. S. Mian, and R. Owens. Sparse approximated nearest points for image set classification. In *CVPR*, 2011.

[6] S. Lyu. Mercer Kernels for Object Recognition with Local Features. Technical report, 2004.

[7] S. Lyu. Mercer Kernels for Object Recognition with Local Features. In *CVPR*, 2005.

[8] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008.

[9] K. Rematas, M. Fritz, and T. Tuytelaars. The pooled NBNN kernel: beyond image-to-class and image-to-image. In *ACCV*, 2012.

[10] J. Sang and C. Xu. Character-based movie summarization. In *ACM Multimedia*, 2010.

[11] B. Settles. Active learning literature survey. Technical report, 2010.

[12] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient sOLver for SVM. In *ICML*, 2007.

[13] J. Sivic, M. Everingham, and A. Zisserman. "Who are you?" – Learning person specific classifiers from video. In *CVPR*, 2009.

[14] M. Tapaswi, M. Bäuml, and R. Stiefelhagen. "Knock! Knock! Who is it?" Probabilistic Person Identification in TV-Series. In *CVPR*, 2012.

[15] A. Vedaldi and A. Zisserman. Sparse kernel approximations for efficient classification and detection. In *CVPR*, 2012.

[16] C. Wallraven, B. Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *ICCV*. IEEE, 2003.

[17] R. Wang, H. Guo, L. Davis, and Q. Dai. Covariance discriminative learning: A natural and efficient approach to image set classification. In *CVPR*, 2012.

[18] R. Wang, S. Shan, X. Chen, and W. Gao. Manifold-Manifold Distance with application to face recognition based on image set. *CVPR*, June 2008.