Incremental Accelerated Kernel Discriminant Analysis

Nikolaos Gkalelis
CERTH-ITI
Thessaloniki, Greece
gkalelis@iti.gr

Vasileios Mezaris
CERTH-ITI
Thessaloniki, Greece
bmezaris@iti.gr

ABSTRACT

In this paper a novel incremental dimensionality reduction (DR) technique called incremental accelerated kernel discriminant analysis (IAKDA) is proposed. Consisting of the eigenvalue decomposition of a relatively small-size matrix and the recursive block Cholesky factorization of the kernel matrix, a nonlinear DR transformation is efficiently computed at each incremental step. Moreover, employing factorization techniques of excellent numerical stability, IAKDA effectively removes data nonlinearities in the low dimensional subspace. Experimental evaluation on various multimedia tasks and datasets confirms that the proposed approach combined with linear support vector machines (LSVMs) offers improved mean average precision (MAP) and provides an impressive training time speedup over batch KDA and also over traditional LSVM and kernel SVM (KSVM).

CCS CONCEPTS

• Information systems → Multimedia and multimodal retrieval; **• Theory of computation → Online learning algorithms**; **• Computing methodologies → Visual content-based indexing and retrieval**; **Dimensionality reduction and manifold learning**; **Kernel methods**;

KEYWORDS

Incremental, kernels, discriminant analysis, dimensionality reduction, machine learning

ACM Reference Format:
Nikolaos Gkalelis and Vasileios Mezaris. 2017. Incremental Accelerated Kernel Discriminant Analysis. In *Proceedings of MM '17*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3123266.3123401

1 INTRODUCTION

Kernel discriminant analysis (KDA) is one of the most popular dimensionality reduction techniques with important applications, among others, in multimedia analysis, computer vision and visualization [1, 2, 4, 13, 16, 22, 23, 31, 42]. This method learns a nonlinear transformation by first applying a nonlinear mapping from the input space to a new feature space and then solving a generalized eigenproblem (GEP) defined by the between- and within-scatter matrices. Despite its success in many challenging problems, its difficulty to scale well with the number of training observations has restricted the widespread use of this technology in today's Big Data problems. This limitation is much more intense in incremental learning problems [37, 53], where new training observations arrive sequentially one at a time, or in larger groups called chunks, as shown in Figure 1. For instance, in video event detection applications [8, 14, 47–49] where usually the videos of the event are scarce and the event detectors are initially trained using only a few of them, the use of KDA to learn a new transformation matrix each time a new positive video example becomes available would have severe implications in the time-efficiency and usefulness of the overall approach.

During the past few years, a quite large variety of incremental linear discriminant analysis (LDA) algorithms have been developed. For instance, in [37], the between- and within-class scatter matrices are updated and their generalized eigenvalue decomposition is computed at every incremental step. A sequential DR method based on a rank-one QR updating algorithm [19] and the adaptive computation of the desired scatter matrices is presented in [55]. An extension of the above approach so that it can deal with chunks of incoming data is provided in [39]. In [56], the projection and total scatter matrices are updated using a variant of the generalized singular value decomposition. Assuming centered data, in [28, 54], the indicator, class-means and projection matrices are updated at each incremental step using a least-square-based algorithm. In [52], the above method is extended for real-time streaming applications. In [9], an adaptive LDA method based on the incremental estimation of the square root of the inverse covariance matrix is proposed. Due to the slow convergence of this approach, in [33] a quasi-Newton

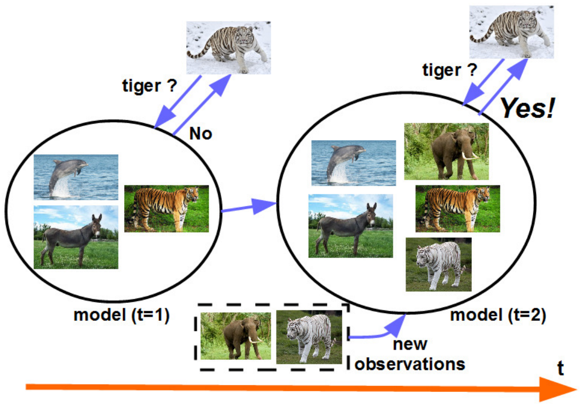

Figure 1: Incremental learning

algorithm is applied to accelerate the estimation of the above matrix. In [15, 32], an explicit cost function and the Newton-Raphson or Gradient Descent algorithms are employed to provide a further speedup of the method presented in [9]. The sufficient set spanning approximation is used in [24] for updating the scatter and linear transformation matrices each time new observations are fetched. This approach is among the most efficient and accurate incremental variants of LDA.

The major drawback of the above methods is that they can not be successfully applied to nonlinear problems. The sequential variant of spectral regression KDA (SRKDA) presented in [6] is one of the few nonlinear incremental variants of LDA in the literature. This method computes a discriminant subspace using the Gram-Schmidt orthogonalization and spectral regression. However, it requires the centering of the data in the feature space, which may significantly increase its computational cost and round-off errors. Moreover, due to the centering operation, the assumption of SRKDA and its incremental variant that the kernel matrix is nonsingular is always violated, breaking down the theoretical development of the method [31, 38].

In [17, 18], accelerated KDA (AKDA) is presented in order to overcome the deficiencies of SRKDA and achieve state-of-the-art training time performance. Specifically, AKDA utilizes a novel matrix factorization and a simultaneous diagonalization framework to efficiently derive the DR transformation matrix without requiring normalized data. However, despite its efficiency this method is not appropriate for incremental learning problems. To this end, incremental AKDA (IAKDA)[1] is proposed here that exploits the framework introduced in [17, 18] to initialize the DR transformation matrix, and the recursive block Cholesky factorization [19] to update this matrix incrementally. The proposed method offers several advantages over the incremental approaches described above, more importantly, it can effectively deal with data nonlinearities and does not require any data normalization. The experimental evaluation on various image and video classification datasets [2, 45, 46] shows that the proposed approach achieves improved accuracy and an impressive speedup over linear and kernel DA, and their incremental variants. Moreover, as shown by the experimental results, the combination of IAKDA with pre-trained convolutional neural networks (CNNs) for feature representation provides excellent classification performance and several orders of magnitude faster training times in comparison to retraining CNNs at each incremental step. In summary, the main contributions of this paper are:

- A novel incremental nonlinear DR method is proposed, which in contrast to the previous approaches deals effectively with nonlinear learning problems and does not require zero mean data normalization.
- As demonstrated by the experimental evaluation, the proposed method achieves state-of-the-art training time performance and improved detection accuracy in various multimedia tasks.

The rest of the paper is structured as follows: The problem formulation is given in Section 2. Section 3 describes the application of traditional batch KDA in the incremental learning problem, while the proposed IAKDA is presented in Section 4. In Section 5, evaluation results on various datasets are provided. Finally, conclusions are drawn in Section 6.

[1] Source code is made publicy available at https://www.iti.gr/~bmezaris/downloads.html

2 PROBLEM FORMULATION

In incremental learning, at each time t the overall dataset $\mathbf{X}_t$ consists of a previously existing dataset, $\check{\mathbf{X}}_t = [\check{\mathbf{x}}_1, \ldots, \check{\mathbf{x}}_{\check{N}_t}]$, and a new dataset $\tilde{\mathbf{X}}_t = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{\tilde{N}_t}]$ containing the observations fetched at time t, where, $\check{\mathbf{x}}_\kappa, \tilde{\mathbf{x}}_\kappa \in \mathbb{R}^L$, L is the dimensionality of the input space $\mathbb{R}^L$, and $\check{N}_t, \tilde{N}_t$ are the number of observations in $\check{\mathbf{X}}_t$ and $\tilde{\mathbf{X}}_t$, respectively. Thus, the total number of observations at time t is $N_t = \check{N}_t + \tilde{N}_t$, and the entire dataset can be represented as

$$\begin{aligned} \mathbf{X}_t &= [\check{\mathbf{X}}_t, \tilde{\mathbf{X}}_t] \\ &= [\check{\mathbf{x}}_1, \ldots, \check{\mathbf{x}}_{\check{N}_t}, \tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{\tilde{N}_t}] \\ &= [\mathbf{x}_1, \ldots, \mathbf{x}_{N_t}]. \end{aligned} \quad (1)$$

We assume that the dataset is annotated, that is, each observation $\mathbf{x}_\kappa$ is paired with a label $y_\kappa \in \{1, \ldots, C_t\}$, where C_t is the number of classes at time t. The target in incremental DR is to compute the DR transformation matrix at time t by exploiting the previously computed quantities at time $t-1$, and the new dataset $\tilde{\mathbf{X}}_t$. This contrasts batch DR learning where the entire set $\mathbf{X}_t$ is used as a whole ignoring the time information.

3 BATCH KDA

Conventional KDA [4, 31, 42], called hereafter batch KDA, is not designed for incremental learning problems. That is, for learning the DR transformation matrix at time t it uses the entire dataset $\mathbf{X}_t$ (1), ignoring any computed quantities at time $t-1$. To deal with nonlinearly-separable classes, the batch KDA utilizes a nonlinear mapping $\boldsymbol{\phi}(\cdot)$ from the input space to the so-called feature space $\mathbb{R}^F$ of dimensionality F, associated with the Mercer kernel $k(\cdot,\cdot)$,

$$\begin{aligned} \boldsymbol{\phi}(\cdot) &: \mathbb{R}^L \mapsto \mathbb{R}^F, \\ k(\cdot,\cdot) &: \mathbb{R}^L \times \mathbb{R}^L \mapsto \mathbb{R}, \end{aligned}$$

so that the inner products in the feature space can be replaced with kernel function evaluations [21, 30]. Given $\mathbf{X}_t$, it then seeks the linear transformation $\mathbf{\Gamma}_t \in \mathbb{R}^{F \times D_t}$, $D_t \ll F$, that maximizes the following criterion

$$\underset{\mathbf{\Gamma}_t}{\operatorname{argmax}} \operatorname{tr}((\mathbf{\Gamma}_t^T \Sigma_{w,t} \mathbf{\Gamma}_t)^\dagger \mathbf{\Gamma}_t^T \Sigma_{b,t} \mathbf{\Gamma}_t), \quad (2)$$

where, $\operatorname{tr}(\mathbf{A})$, $\mathbf{A}^T$ and $\mathbf{A}^\dagger$ denote trace, transpose and pseudoinverse of matrix $\mathbf{A}$, respectively. In (2), $\Sigma_{b,t}$ and $\Sigma_{w,t}$ are the between- and within-class scatter matrices at time t,

$$\Sigma_{b,t} = \sum_{i=1}^{C_t} N_{i,t} (\boldsymbol{\mu}_{i,t} - \boldsymbol{\mu}_t)(\boldsymbol{\mu}_{i,t} - \boldsymbol{\mu}_t)^T, \quad (3)$$

$$\Sigma_{w,t} = \sum_{i=1}^{C_t} \sum_{\substack{\kappa=1 \\ y_\kappa == i}}^{N_t} (\boldsymbol{\phi}_\kappa - \boldsymbol{\mu}_{i,t})(\boldsymbol{\phi}_\kappa - \boldsymbol{\mu}_{i,t})^T, \quad (4)$$

and $\boldsymbol{\phi}_\kappa$, $N_{i,t}$, $\boldsymbol{\mu}_{i,t}$, $\boldsymbol{\mu}_t$, are the κth observation in $\mathbb{R}^F$, the number of observations and estimated sample mean of class i, and sample mean of the entire dataset, respectively. The optimization problem in (2) is equivalent to finding the nonzero eigenpairs (NZEP) of the following generalized eigenproblem (GEP)

$$\Sigma_{b,t}\Gamma_t = \Sigma_{w,t}\Gamma_t\Lambda_t, \tag{5}$$

where, Γ_t's columns are the eigenvectors of the matrix pencil ($\Sigma_{b,t}$, $\Sigma_{w,t}$), $\Lambda_t \in \mathbb{R}^{D_t \times D_t}$ is the diagonal eigenvalue matrix whose diagonal elements are sorted in descending order, and $D_t = \mathrm{rank}(\Sigma_{b,t})$. To avoid working directly in $\mathbb{R}^F$, which may be of very high or infinity dimensionality, the transformation matrix is expressed as $\Gamma_t = \Phi_t\Psi_t$, where,

$$\Phi_t = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_{N_t}] \in \mathbb{R}^{F \times N_t}$$

are the mapped training data, and $\Psi_t \in \mathbb{R}^{N_t \times D_t}$ is the expansion coefficient matrix. The following GEP is then considered

$$\mathbf{S}_{b,t}\Psi_t = \mathbf{S}_{w,t}\Psi_t\Lambda_t, \tag{6}$$

which under mild condition [34] is equivalent to the one in (5). That is, we now need to identify the NZEP (Ψ_t, Λ_t) of the matrix pencil ($\mathbf{S}_{b,t}, \mathbf{S}_{w,t}$), where, $\mathbf{S}_{b,t}$, $\mathbf{S}_{w,t}$, are the kernel matrices associated with the between- and within-class variability

$$\mathbf{S}_{b,t} = \sum_{i=1}^{C_t} N_{i,t}(\boldsymbol{\eta}_{i,t} - \boldsymbol{\eta}_t)(\boldsymbol{\eta}_{i,t} - \boldsymbol{\eta}_t)^T, \tag{7}$$

$$\mathbf{S}_{w,t} = \sum_{i=1}^{C_t} \sum_{\substack{\kappa=1 \\ y_\kappa == i}}^{N_t} (\mathbf{k}_\kappa - \boldsymbol{\eta}_{i,t})(\mathbf{k}_\kappa - \boldsymbol{\eta}_{i,t})^T, \tag{8}$$

$\boldsymbol{\eta}_{i,t} = \mathbf{K}_{i,t}\tilde{\mathbf{1}}_{N_{i,t}}$, $\boldsymbol{\eta}_t = \mathbf{K}_{t,t}\tilde{\mathbf{1}}_{N_t}$, $\tilde{\mathbf{1}}_{N_{i,t}} = \frac{1}{N_{i,t}}\mathbf{1}_{N_{i,t}}$, $\tilde{\mathbf{1}}_{N_t} = \frac{1}{N_t}\mathbf{1}_{N_t}$, $\mathbf{1}_{N_{i,t}} \in \mathbb{R}^{N_{i,t}}$ is the all-one vector, $\mathbf{K}_{i,t} = \Phi_t^T\Phi_{i,t}$, $\Phi_{i,t} \in \mathbb{R}^{F \times N_{i,t}}$ contains the observations of the ith class, $\mathbf{k}_\kappa = [k(\mathbf{x}_1, \mathbf{x}_\kappa), \dots, k(\mathbf{x}_{N_t}, \mathbf{x}_\kappa)]^T$ is the kernel vector associated with the κth training observation, and $\mathbf{K}_{t,t}$ is the kernel matrix of the training observations,

$$\mathbf{K}_{t,t} = \Phi_t^T\Phi_t = [\mathbf{k}_1, \dots, \mathbf{k}_{N_t}]. \tag{9}$$

We should note that $\Sigma_{w,t}$ and $\mathbf{S}_{w,t}$ are often replaced by their total scatter matrix counterparts,

$$\Sigma_{m,t} = \sum_{n=1}^{N_t}(\boldsymbol{\phi}_\kappa - \boldsymbol{\mu}_t)(\boldsymbol{\phi}_\kappa - \boldsymbol{\mu}_t)^T, \tag{10}$$

$$\mathbf{S}_{m,t} = \sum_{\kappa=1}^{N_t}(\mathbf{k}_\kappa - \boldsymbol{\eta}_t)(\mathbf{k}_\kappa - \boldsymbol{\eta}_t)^T, \tag{11}$$

respectively. Typically, the GEP in (6) is solved using techniques that compute the simultaneous diagonalization of $\mathbf{S}_{b,t}$ and $\mathbf{S}_{w,t}$ (or $\mathbf{S}_{m,t}$) [19, 38]. Given the identified linear transformation Ψ_t, the projection $\mathbf{z} \in \mathbb{R}^{D_t}$ of a test vector $\mathbf{x} \in \mathbb{R}^L$ can be then computed using $\mathbf{z} = \Gamma_t^T\boldsymbol{\phi} = \Psi_t^T\Phi_t^T\boldsymbol{\phi} = \Psi_t^T\mathbf{k}$, where

$$\mathbf{k} = [k(\mathbf{x}_1, \mathbf{x}), \dots, k(\mathbf{x}_{N_t}, \mathbf{x})]^T.$$

We observe that using the batch KDA approach described above, the various scatter matrices in (7), (8), (or (11)), and the GEP in (6) need to be computed for each point in time t that a new chunk of observations is fetched. Taking into account that the computational cost of KDA at each time period t is approximately $40N_t^3$ [17, 18], we can easily conclude that the use of this method in incremental learning applications is impractical.

4 PROPOSED IAKDA

Given $\mathbf{X}_t$ (1) at time t, IAKDA computes the coefficient matrix $\mathbf{V}_t$ satisfying the following simultaneous diagonalization [17, 18]

$$\mathbf{W}_t^T\Sigma_{b,t}\mathbf{W}_t = \mathbf{I}_{C_t-1}, \tag{12}$$

$$\mathbf{W}_t^T\Sigma_{w,t}\mathbf{W}_t = \mathbf{0}_{C_t-1}, \tag{13}$$

$$\mathbf{W}_t^T\Sigma_{m,t}\mathbf{W}_t = \mathbf{I}_{C_t-1}, \tag{14}$$

where, $\mathbf{0}_{C_t-1} \in \mathbb{R}^{C_t-1 \times C_t-1}$ is the all-zero matrix, and $\mathbf{W}_t = \Phi_t\mathbf{V}_t$. That is, the computed $\mathbf{W}_t$ satisfies the KDA optimization criterion presented in (2) [17, 18, 31, 38]. IAKDA consists of an initialization and an incremental updating part, as described in the following subsections.

4.1 Initialization

The initialization step of IAKDA is based on AKDA [17, 18]. Let $\mathbf{X}_t = [\mathbf{x}_1, \dots, \mathbf{x}_{N_t}]$, where $t = 1$, be the initial dataset fetched by the learning system. Following AKDA, the respective scatter matrices $\Sigma_{b,t}$ (3), $\Sigma_{w,t}$ (4) and $\Sigma_{m,t}$ (10), can be factorized as

$$\Sigma_{b,t} = \Phi_t\mathbf{C}_{b,t}\Phi_t^T,$$

$$\Sigma_{w,t} = \Phi_t\mathbf{C}_{w,t}\Phi_t^T,$$

$$\Sigma_{m,t} = \Phi_t\mathbf{C}_{m,t}\Phi_t^T,$$

where,

$$\mathbf{C}_{b,t} = \mathbf{R}_t\mathbf{N}_t^{-1/2}\mathbf{O}_t\mathbf{N}_t^{-1/2}\mathbf{R}_t^T \tag{15}$$

$$\mathbf{C}_{w,t} = \mathbf{I}_{N_t} - \mathbf{R}_t\mathbf{N}_t^{-1}\mathbf{R}_t^T,$$

$$\mathbf{C}_{m,t} = \mathbf{I}_{N_t} - \frac{1}{N_t}\mathbf{J}_{N_t}$$

$$\mathbf{n}_t = [N_{1,t}, \dots, N_{C_t,t}]^T, \tag{16}$$

$$\mathbf{N}_t = \mathrm{diag}(N_{1,t}, \dots, N_{C_t,t}), \tag{17}$$

$$[\mathbf{R}_t]_{n,i} = \begin{cases} 1 & \text{if } y_\kappa == i, \\ 0 & \text{else,} \end{cases} \tag{18}$$

and, $\mathbf{J}_{N_t}, \mathbf{I}_{N_t} \in \mathbb{R}^{N_t \times N_t}$ are the all-one matrix and identity matrices. Above, $\mathbf{n}_t$, $\mathbf{N}_t$ are the so-called strength vector and matrix, $[\mathbf{R}_t]_{n,i}$ is the element in the nth row and ith column of the class indicator matrix $\mathbf{R}_t \in \mathbb{R}^{N_t \times C_t}$, and $\mathbf{N}_t^{1/2} = \mathrm{diag}(\sqrt{N_{1,t}}, \dots, \sqrt{N_{C_t,t}})$. Moreover, in (15), $\mathbf{O}_t$ is the so-called core matrix, defined as

$$\mathbf{O}_t = \mathbf{I}_{C_t} - \frac{\mathbf{n}_t^{1/2}\mathbf{n}_t^{T/2}}{\mathbf{n}_t^{T/2}\mathbf{n}_t^{1/2}}, \tag{19}$$

where $\mathbf{n}_t^{T/2}$ denotes the transpose of $\mathbf{n}_t^{1/2} = [\sqrt{N_{1,t}}, \dots, \sqrt{N_{C_t,t}}]^T$. Let $\Theta_t \in \mathbb{R}^{N_t \times D_t}$ be the matrix defined as follows

$$\Theta_t = \mathbf{R}_t\mathbf{N}_t^{-1/2}\Xi_t, \tag{20}$$

where the columns of $\Xi_t \in \mathbb{R}^{C_t \times C_t-1}$ are the eigenvectors of $\mathbf{O}_t$ corresponding to its nonzero eigenvalues. As shown in [17, 18], Θ_t provides the simultaneous diagonalization of $\mathbf{C}_{b,t}$, $\mathbf{C}_{w,t}$ and $\mathbf{C}_{m,t}$. The expansion coefficient matrix $\mathbf{V}_t$ can then be computed by solving the following linear system

$$\mathbf{K}_{t,t}\mathbf{V}_t = \Theta_t. \tag{21}$$

It can be verified that the respective transformation $\mathbf{W}_t$ provides the desired simultaneous diagonalization presented in (12), (13), (14).

For more details on the above mathematical derivations the interested reader is refereed to [17, 18]. The kernel matrix $\mathbf{K}_{t,t}$ is always symmetric positive definite or symmetric positive semidefinite. To this end, its Cholesky factorization is used to efficiently solve the linear system in (21). The overall procedure for the initialization of IAKDA is summarized in Algorithm 1. We should note that based on AKDA, the initialization step of IAKDA is very efficient offering an approximately $O(N_t^3/3)$ computational complexity dominated by the Cholesky factorization of $\mathbf{K}_{t,t}$ [17–19]. More importantly, as we see in the next subsection, the above framework facilitates the design of an incremental updating step, offering a further speedup over batch KDA as well as AKDA.

Algorithm 1 IAKDA: Initialization

```
Input: $\mathbf{X}_1, k(\cdot,\cdot)$
Output: $\mathbf{V}_1, \mathbf{X}_1, \mathbf{R}_1, \mathbf{n}_1, \mathbf{N}_1, \mathbf{K}_{1,1}, \mathbf{L}_1$
1: Compute the elementary matrices $\mathbf{R}_1$ (18), $\mathbf{n}_1$ (16), $\mathbf{N}_1$ (17)
2: Compute $\mathbf{O}_1$ (19) and its eigenvector matrix $\Xi_1$
3: Compute the eigenvector matrix $\mathbf{\Theta}_1$ (20)
4: Compute $\mathbf{K}_{1,1}$ (9) and its Cholesky factor $\mathbf{L}_1$
5: Solve the linear system in (21) to obtain $\mathbf{V}_1$
```

4.2 Incremental updating

As defined in (1), let $\check{\mathbf{X}}_t = [\check{\mathbf{x}}_1, \ldots, \check{\mathbf{x}}_{\check{N}_t}]$ and $\tilde{\mathbf{X}}_t = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{\tilde{N}_t}]$ be the existing and new dataset at time t, respectively. Thus, the overall dataset at time t is formed as follows

$$\mathbf{X}_t = [\check{\mathbf{X}}_t, \tilde{\mathbf{X}}_t].$$

In AKDA the overall dataset $\mathbf{X}_t$ is used to obtain $\mathbf{V}_t$ as shown in the initialization step above and described in more detail in [17, 18]. In contrary, IAKDA computes $\mathbf{V}_t$ very efficiently using the precomputed arrays at $t-1$ and the new dataset $\tilde{\mathbf{X}}_t$, as explained in the following. The indicator matrix and the strength vector and matrix can be efficiently updated as follows

$$\mathbf{R}_t = [\check{\mathbf{R}}_t^T, \tilde{\mathbf{R}}_t^T]^T, \quad (22)$$

$$\mathbf{n}_t = \check{\mathbf{n}}_t + \tilde{\mathbf{n}}_t, \quad (23)$$

$$\mathbf{N}_t = \check{\mathbf{N}}_t + \tilde{\mathbf{N}}_t, \quad (24)$$

where, $\check{\mathbf{R}}_t$, $\check{\mathbf{n}}_t$, $\check{\mathbf{N}}_t$ and $\tilde{\mathbf{R}}_t$, $\tilde{\mathbf{n}}_t$, $\tilde{\mathbf{N}}_t$ are the indicator matrix, strength vector and strength matrix associated with $\check{\mathbf{X}}_t$ and $\tilde{\mathbf{X}}_t$, respectively. In case that not all existing classes $\check{C}_t$ are represented in $\tilde{\mathbf{X}}_t$, or when new classes are introduced in $\tilde{\mathbf{X}}_t$, the corresponding arrays in (22), (23) and (24) are padded with zeros accordingly, in order to allow the execution of the respective matrix operations. Utilizing the updated $\mathbf{n}_t$, the core matrix $\mathbf{O}_t$ can be computed using (19) and subsequently its eigenvector matrix Ξ_t can be easily retrieved. Similarly, utilizing the updated $\mathbf{R}_t$, $\mathbf{N}_t$ and Ξ_t, the eigenvector matrix $\mathbf{\Theta}_t$ is obtained using (20). The kernel matrix $\mathbf{K}_{t,t} = \mathbf{\Phi}_t^T \mathbf{\Phi}_t$ can be represented as

$$\mathbf{K}_{t,t} = \begin{bmatrix} \check{\mathbf{K}}_{t,t} & \breve{\mathbf{K}}_{t,t}^T \\ \breve{\mathbf{K}}_{t,t} & \tilde{\mathbf{K}}_{t,t} \end{bmatrix}, \quad (25)$$

where, $\mathbf{\Phi}_t = [\check{\mathbf{\Phi}}_t, \tilde{\mathbf{\Phi}}_t]$, $\check{\mathbf{\Phi}}_t$, $\tilde{\mathbf{\Phi}}_t$ are the feature space representation of $\check{\mathbf{X}}_t$ and $\tilde{\mathbf{X}}_t$, respectively, $\check{\mathbf{K}}_{t,t} = \check{\mathbf{\Phi}}_t^T \check{\mathbf{\Phi}}_t$ is the existing kernel matrix, already computed in time $t-1$, $\breve{\mathbf{K}}_{t,t} = \tilde{\mathbf{\Phi}}_t^T \check{\mathbf{\Phi}}_t$, and $\tilde{\mathbf{K}}_{t,t} = \tilde{\mathbf{\Phi}}_t^T \tilde{\mathbf{\Phi}}_t$. Based on the above formulation, the Cholesky factor update is computed using a block Cholesky framework [19] as explained in the following. Suppose that the lower triangular Cholesky factor $\mathbf{L}_t$ of $\mathbf{K}_{t,t}$ is expressed in block format as

$$\mathbf{L}_t = \begin{bmatrix} \mathbf{L}_{1,t} & \breve{\mathbf{0}}_t \\ \mathbf{L}_{2,t} & \mathbf{L}_{3,t} \end{bmatrix}, \quad (26)$$

where, the blocks $\mathbf{L}_{1,t} \in \mathbb{R}^{\check{N}_t \times \check{N}_t}$, $\mathbf{L}_{2,t} \in \mathbb{R}^{\tilde{N}_t \times \check{N}_t}$, $\mathbf{L}_{3,t} \in \mathbb{R}^{\tilde{N}_t \times \tilde{N}_t}$ are lower triangular matrices needed to be identified, and $\breve{\mathbf{0}}_t$ is the $\check{N}_t \times \tilde{N}_t$ all-zero matrix. Using (26), $\mathbf{K}_{t,t}$ can be expressed as

$$\mathbf{K}_{t,t} = \mathbf{L}_t \mathbf{L}_t^T = \begin{bmatrix} \mathbf{L}_{1,t}\mathbf{L}_{1,t}^T & \mathbf{L}_{1,t}\mathbf{L}_{2,t}^T \\ \mathbf{L}_{2,t}\mathbf{L}_{1,t}^T & \mathbf{L}_{2,t}\mathbf{L}_{2,t}^T + \mathbf{L}_{3,t}\mathbf{L}_{3,t}^T \end{bmatrix}. \quad (27)$$

Equating blocks between (25) and (27) we get

$$\check{\mathbf{K}}_{t,t} = \mathbf{L}_{1,t}\mathbf{L}_{1,t}^T, \quad (28)$$

$$\breve{\mathbf{K}}_{t,t} = \mathbf{L}_{2,t}\mathbf{L}_{1,t}^T, \quad (29)$$

$$\tilde{\mathbf{K}}_{t,t} = \mathbf{L}_{2,t}\mathbf{L}_{2,t}^T + \mathbf{L}_{3,t}\mathbf{L}_{3,t}^T. \quad (30)$$

From (28) we observe that $\mathbf{L}_{1,t}$ is identical to the Cholesky factor $\check{\mathbf{L}}_t$ of $\check{\mathbf{K}}_{t,t}$, which has been already computed in the previous time period

$$\mathbf{L}_{1,t} = \check{\mathbf{L}}_t.$$

The next block of the Cholesky factor can then be easily retrieved by solving the triangular system in (29) yielding

$$\mathbf{L}_{2,t} = \breve{\mathbf{K}}_{t,t}\check{\mathbf{L}}_t^{-T},$$

where $\mathbf{A}^{-T}$ denotes the inverse of any matrix $\mathbf{A}^T$. Finally, by the following reformulation of (30),

$$\tilde{\mathbf{K}}_{t,t} - \breve{\mathbf{K}}_{t,t}\check{\mathbf{K}}_{t,t}^{-1}\breve{\mathbf{K}}_{t,t}^T = \mathbf{L}_{3,t}\mathbf{L}_{3,t}^T,$$

we observe that $\mathbf{L}_{3,t}$ is the Cholesky factor of $\tilde{\mathbf{K}}_{t,t} - \breve{\mathbf{K}}_{t,t}\check{\mathbf{K}}_{t,t}^{-1}\breve{\mathbf{K}}_{t,t}^T$. Using the computed $\mathbf{\Theta}_t$ and $\mathbf{L}_t$, the linear transformation $\mathbf{V}_t$ is then retrieved by solving the following two triangular linear systems

$$\mathbf{L}_t \mathbf{Y}_t = \mathbf{\Theta}_t, \quad (31)$$

$$\mathbf{L}_t \mathbf{V}_t = \mathbf{Y}_t. \quad (32)$$

The incremental updating part of IAKDA is depicted in Algorithm 2. In summary, we see that $\mathbf{R}_t$ (22), $\mathbf{n}_t$ (23), $\mathbf{N}_t$ (24), $\mathbf{K}_{t,t}$ (25) and $\mathbf{L}_t$ (26) are computed by fully exploiting their respective counterparts in the previous period of time, thus, achieving a significant speedup over the application of KDA or AKDA in the incremental learning problem. We also observe that in overall IAKDA has very good numerical properties due to the fact that it consists of a

Algorithm 2 IAKDA: Incremental updating

```
Input: $\check{\mathbf{X}}_t, \tilde{\mathbf{X}}_t, \check{\mathbf{R}}_t, \check{\mathbf{n}}_t, \check{\mathbf{N}}_t, \check{\mathbf{K}}_{t,t}, \check{\mathbf{L}}_t, k(\cdot,\cdot)$
Output: $\mathbf{V}_t, \mathbf{X}_t, \mathbf{R}_t, \mathbf{n}_t, \mathbf{N}_t, \mathbf{K}_{t,t}, \mathbf{L}_t$
1: Update elementary matrices $\mathbf{R}_t$ (22), $\mathbf{n}_t$ (23), $\mathbf{N}_t$ (24)
2: Compute $\mathbf{O}_t$ (19) and its eigenvector matrix $\Xi_t$
3: Compute the eigenvector matrix $\mathbf{\Theta}_t$ (20)
4: Update the kernel matrix $\mathbf{K}_{t,t}$ (25)
5: Update the Cholesky factor $\mathbf{L}_t$ (26)
6: Solve the two triangular systems in (31), (32) to obtain $\mathbf{V}_t$
```

few elementary matrix operations and very stable decomposition algorithms.

5 EXPERIMENTAL EVALUATION

The proposed methods are evaluated using various datasets for the tasks of object, concept and event detection in images and videos, as described in the next subsections.

5.1 Datasets

For the experimental evaluation the following 7 datasets are used: a) *AwA* [25]: provides images matching the 50 animal categories in Osherson's animal/attribute matrix, b) *Ayahoo* [12]: consists of various object and concept images collected using the Yahoo! image search engine, c) *Caltech101* [27]: contains images belonging to a wide variety of categories, such as airplane, chair and crocodile, d) *Eth80* [26]: consists of 8 object classes, recorded from 41 different views, e) *Imagenet* [10]: this is a large-scale concept detection dataset, f) *Med10* [20, 36]: contains videos belonging to one of 3 real-world events or to the "rest-of-the-world" event, g) *Office* [43]: provides images of various resolution from the Amazon.com and an office environment captured using a webcam and a digital LSR camera. From each of the 6 latter datasets 3 images are depicted in Figure 2 (representative images from AwA dataset are not shown due to copyright reasons). In the med10 dataset, videos are represented in the input space $\mathbb{R}^{101376}$ using the improved dense trajectory features [2, 51]. For the rest of the datasets, the image descriptors corresponding to the 4096 neurons of DeCAF's 6-th layer [11] provided in [45, 46] are used.

5.2 Experimental setup

For the evaluation in the task of event detection, the med10 dataset is already divided to 1744 training and 1742 testing videos. For the rest of the datasets, we randomly selected 100 observations from each class to create a training set, and used the rest of the observations to form the testing set. Moreover, classes of less than 200 observations were equally divided to training and testing sets. The resulted partitions for each dataset are shown in Table 1. Furthermore, each training set is divided into 10 equal partitions, where each partition contains the same number of observations per class. This is done in order to simulate a streaming application, where at each period of time a new chunk of observations is received.

During the evaluation, IAKDA is compared with the incremental LDA (ILDA) presented in [24], and batch versions of LDA, KDA and the state-of-the-art AKDA [17, 18]. For IAKDA our Matlab implementation is used. For ILDA the code provided in [24] is utilized, for LDA and KDA the efficient Matlab functions provided in [5, 6] are exploited, while for AKDA the respective Matlab implementation of [17, 18] is used. For their comparison in the various tasks, the DR methods are further combined with a linear support vector machine (LSVM) [50], which learns the desired target classes in the discriminant subspace. The above methods are also compared with batch versions of LSVM and KSVM directly applied in the input space. For both LSVM and KSVM the library provided in [7] is exploited. The Gaussian kernel is utilized for IAKDA, AKDA, KDA and KSVM. At time period t, the batch LDA, KDA, AKDA, LSVM and KSVM are trained using the overall set $\mathbf{X}_t$, consisting of the existing set $\check{\mathbf{X}}_t$ and the new set of observations $\tilde{\mathbf{X}}_t$, as described for the batch KDA in Section 3. The different approaches are optimized using 3-fold cross-validation (CV), where at each fold the training set is randomly split to 30% learning set and 70% validation set. During the CV procedure, the kernel parameter γ of the Gaussian kernel $\exp(-\gamma\|\mathbf{x}_\kappa - \mathbf{x}_\nu\|^2)$ is searched in $\{0.01, 0.1, 0.6\} \cup \{1, 1.5, \ldots, 7\}$, while for the SVM penalty term the following parameter space is used $\{0.1, 1, 10, 100\}$.

The performance of the mth method at each dataset and time period is measured using the MAP measure, $\bar{\varrho}_m = \frac{1}{C}\sum_{i=1}^{C} \varrho_{m,i}$, and the training time speedup over KDA, $\bar{\theta}_m = \theta_{KDA}/\theta_m$, where $\varrho_{m,i}$ is the average precision [41] of the mth method at the ith class, and θ_m is the training time of method m along C classes in the dataset. In our evaluation, for the med10 dataset all the $C = 3$ event classes are used. For the rest of the datasets, only the first $C = 10$ classes from each one are considered, while the observations belonging to the rest of the classes are regarded as instances of the rest-of-the-world class. Moreover, for the training time we consider only the time for building each classifier with the parameters fixed to the optimal ones derived using the CV procedure, i.e., the CV time is excluded. The evaluation was performed using Intel i7 3770K@3.5Ghz CPU, 32 GB RAM workstations with 64-bit Windows 7.

5.3 Results

The evaluation results in terms of MAP and training time speedup over KDA for each method and along all time periods are shown in Figures 3, 5 and 6. The MAP and training time speedup for the 1st, 5th and 10th time period (i.e. for the first 10th, half and the whole training dataset) are recorded in Tables 2 and 3, respectively.

Table 1: Train/test observations per dataset

	AwA	*ayahoo*	*caltech101*	*eth80*	*imagenet*	*office*
# train	4941	988	3539	1680	11762	2075
# test	25792	1249	5192	1600	153836	2039

Table 2: MAP rates at t = 1, 5, 10

	LDA	*ILDA*	*LSVM*	*KDA*	*AKDA*	*IAKDA*	*KSVM*
AwA − 1	26.3%	26.3%	34.5%	37.6%	**39.5%**	39.1%	38.2%
AwA − 5	25.6%	23.1%	47.2%	52.4%	56.3%	**56.4%**	53.7%
AwA − 10	28.8%	27.2%	52.7%	57.4%	63.3%	**63.5%**	60.4%
ayahoo − 1	67%	66.9%	70.4%	70.7%	**71.3%**	70.9%	70.7%
ayahoo − 5	71.3%	71.3%	79.6%	79.4%	**81.7%**	81.6%	81.8%
ayahoo − 10	71.5%	72.1%	84.2%	82.6%	87.1%	**87.7%**	85.4%
caltech101 − 1	29.2%	28.9%	**59.5%**	59%	57.5%	58%	57.4%
caltech101 − 5	62.6%	62%	73.3%	**76.8%**	76.1%	76%	74.6%
caltech101 − 10	60.3%	69.1%	76.2%	80.7%	82.6%	**82.7%**	78%
eth80 − 1	25.4%	11.9%	68.2%	63.2%	62.7%	62.3%	**68.7%**
eth80 − 5	76%	79.6%	83.8%	74.9%	**85.7%**	84.8%	83.6
eth80 − 10	73.2%	83.6%	84.4%	77.8%	**87.5%**	86.8%	84.3
imagenet − 1	12.6%	13.8%	29.4%	32.1%	**37%**	36.6%	34.3%
imagenet − 5	12.7%	22.3%	46.6%	55.8%	**57.9%**	57.6%	51.1%
imagenet − 10	31.7%	31.8%	49.7%	60.3%	63.2%	**63.5%**	57.2%
med10 − 1	30.7%	30.8%	30.2%	30.5%	**30.6%**	29.8%	29.8%
med10 − 5	**52%**	**52%**	51.9%	48.5%	51.9%	51.9%	50.1%
med10 − 10	56.4%	55.9%	56.2%	54.8%	**57.6%**	**57.6%**	54.2%
office − 1	60.1%	60.3%	59.9%	57.1%	**64.4%**	63.3%	6.34%
office − 5	69.9%	66%	74.9%	69.9%	**79%**	78.9%	77.4%
office − 10	66.2%	76%	80.5%	74.9%	85.3%	**85.5%**	83.2%

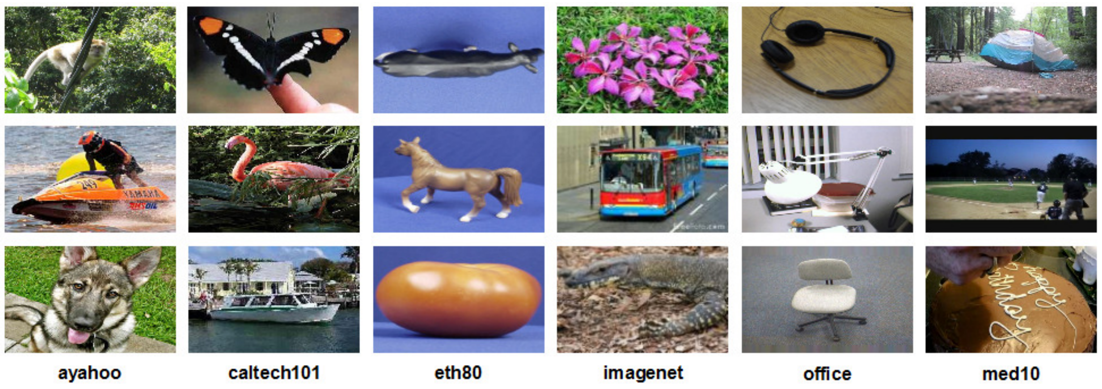

Figure 2: Images from 6 datasets used in the evaluation

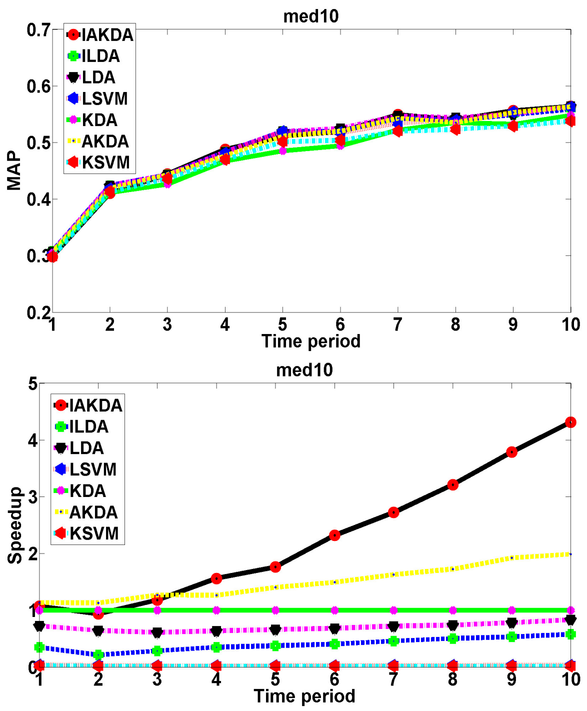

Figure 3: MAP and speedup rates on med10

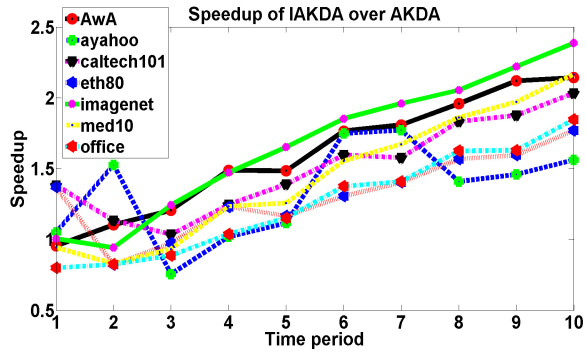

Figure 4: Speedup of IAKDA over AKDA in seven datasets

Table 3: Speedup rates at t = 1, 5, 10

	LDA	*ILDA*	*LSVM*	*KDA*	*AKDA*	*IAKDA*	*KSVM*
AwA – 1	2.1	1	0.3	1	**4.3**	4.1	0.05
AwA – 5	5.4	2	1.4	1	21.8	**32.3**	0.2
AwA – 10	9.9	7	3.2	1	44.1	**94.6**	0.4
ayahoo – 1	1.2	1	0.3	1	**1.4**	**1.4**	0.2
ayahoo – 5	2.7	1.4	0.2	1	5.4	**6**	0.09
ayahoo – 10	2.8	1.4	0.3	1	7.6	**11.8**	0.1
caltech101 – 1	1.9	0.9	0.3	1	1.9	**2.2**	0.1
caltech101 – 5	5.1	1.9	1.8	1	16.6	**23.1**	0.6
caltech101 – 10	6.3	4.1	4.7	1	33.7	**68.6**	1.5
eth80 – 1	1.8	0.8	0.5	1	0.8	**1**	0.3
eth80 – 5	3.2	1.6	1.2	1	7.6	**8.8**	0.8
eth80 – 10	4.9	4.3	2.7	1	16.4	**29**	2
imagenet – 1	3.3	1.3	0.9	1	10.2	**10.3**	0.1
imagenet – 5	59.3	5.3	5.8	1	47.6	**78.5**	0.4
imagenet – 10	**310.6**	41.1	13.4	1	81.8	195.3	0.8
med10 – 1	0.7	0.4	0.04	1	**1.1**	**1.1**	0.02
med10 – 5	0.7	0.4	0.02	1	1.4	**1.8**	0.01
med10 – 10	0.8	0.6	0.03	1	2	**4.3**	0.01
office – 1	1.5	0.8	0.3	1	**2.4**	2	0.2
office – 5	3.5	1.7	0.8	1	10.4	**12**	0.5
office – 10	5.9	2.9	1.9	1	22.2	**40.9**	1.3

Finally, the speedup diagrams depicting the speedup achieved from IAKDA over AKDA along all time periods in the different datasets are provided in Figure 4. From the obtained results we conclude the following:

i) In the majority of the datasets and time periods, IAKDA and AKDA provide the best performance in terms of MAP. For example, in the two larger datasets, AwA and imagenet, at the 10th time period, IAKDA attains a MAP gain over the third best approach of more than 3%. This is due to the very good numerical properties of both methods, which in comparison to the other DR methods, consist of few elementary matrix operations and very stable decomposition algorithms.

ii) In AwA, ayahoo, imagenet, and to a smaller extent in caltech101, we observe that the kernel DR methods (KDA, AKDA and

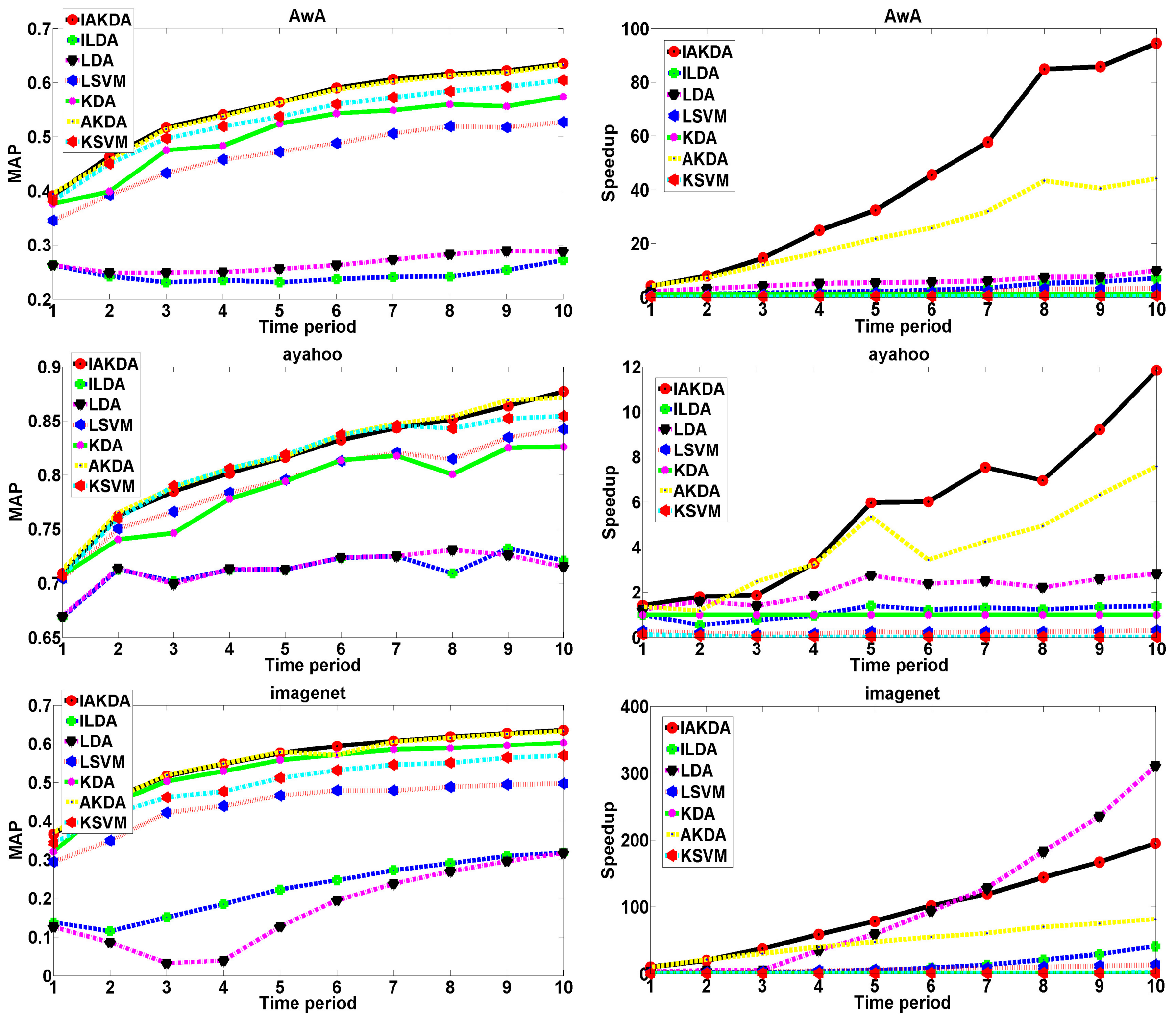

Figure 5: MAP and speedup rates on AwA, ayahoo and imagenet

IAKDA) as well as the LSVM and KSVM applied directly in the input space, achieve a large MAP gain over the linear DR approaches (LDA, ILDA). This shows that the classification tasks in the above datasets are nonlinear, and, thus, the linear DR approaches fail to discover an adequate linear discriminant subspace. The same is true for the office and eth80 datasets, however, with the difference that now KDA underperforms as well. We observe that in this case the number of observations is relatively low, and therefore the small sample size problem is quite intense. In consequence, the KDA scatter matrices are severely ill-posed, which explains the poor performance of this approach in these two datasets. Finally, in the med10 we observe that all methods have quite similar MAP performance. To this end, we conclude that the use of improved dense trajectories has effectively removed nonlinearities in this dataset.

iii) We observe that IAKDA and AKDA attain the best and second best training time performance in the majority of the datasets. Specifically, only LDA in the imagenet dataset provides a better performance after the 7th incremental step, which is expected due to the fact that for large-scale datasets (i.e. when $N >> L$) such as this one, LDA has a $O(L^3)$ complexity.

iv) An impressive training time speedup of IAKDA over the batch KDA is attained in all datasets. For instance, IAKDA is approximately 195, 95 and 69 times faster than KDA in the imagenet, AwA and caltech101 datasets, respectively. As shown in the various figures, this speedup is the aggregation of the speedup offered by the base method (AKDA) plus the additional speedup achieved by the proposed incremental variant (IAKDA).

v) As shown in Figure 4, a linear speedup of IAKDA over its base method, AKDA, is attained. For instance, in the majority of the datasets at the 10th incremental step, IAKDA is already 2 times faster than AKDA. Considering the impressive training time efficiency of the base AKDA method [17, 18], this is a considerable further improvement. Moreover, we observe that this improvement

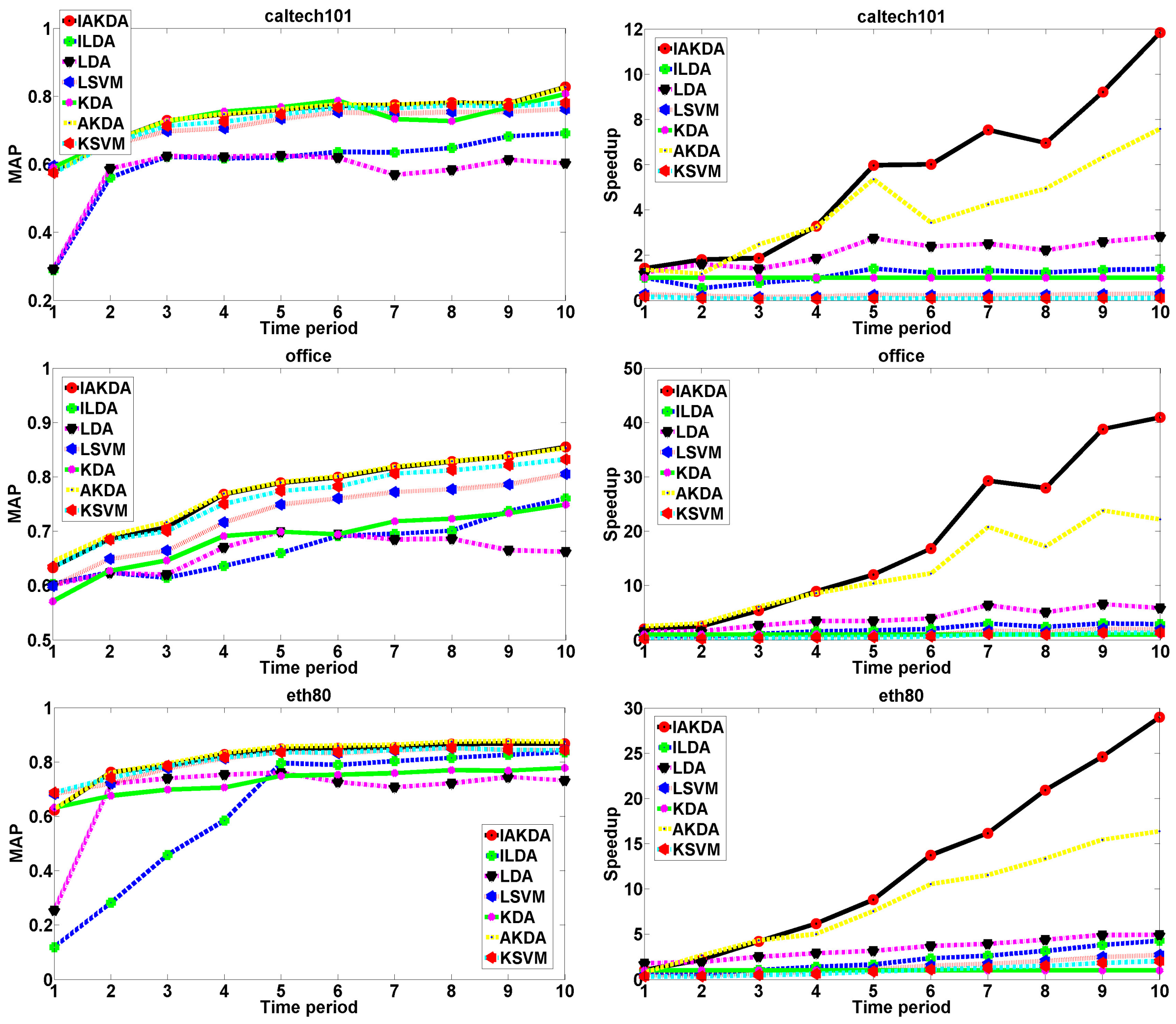

Figure 6: MAP and speedup rates on calthech101, office and eth80

is achieved without sacrificing the detection performance at any of the datasets.

vi) In comparison to deep learning, the proposed approach, i.e. the combination of IAKDA with pre-trained CNNs, is several orders of magnitude faster and it can be applied more efficiently to incremental learning problems. For instance, in our internal evaluations the computation of the initialization step for learning the TRECVID SIN 2013 concepts [3] required a few minutes, versus several hours for fine-tuning the pre-trained GoogleNet [44]. Moreover, preliminary results have shown that using the weights of pre-trained CNNs together with KDA-class algorithms for feature representation and LSVMs for classification outperforms the use of CNNs in the tasks of event and concept detection [1, 29].

Finally, we should note that the most intensive parts of IAKDA can be parallelized using appropriate computing architectures, as for instance is shown in [1, 2, 19], providing even greater training time efficiency.

6 CONCLUSIONS

In this paper, IAKDA was proposed that exploits the simultaneous diagonalization framework presented in [17, 18] and a variant of the block recursive Cholesky factorization to devise an incremental learning algorithm for dimensionality reduction. The experimental evaluation on various image/video datasets demonstrated the effectiveness of IAKDA in both learning time and classification accuracy. Future work includes the investigation of techniques for the efficient simultaneous optimization of IAKDA and CNNs [35] and the use of appropriate fine-tuning approaches, similarly to [40], to further improve performance.

ACKNOWLEDGMENTS

This work was supported by the EU's Horizon 2020 research and innovation programme under grant agreement H2020-732665 EMMA.

REFERENCES

[1] S. Arestis-Chartampilas, N. Gkalelis, and V. Mezaris. 2015. GPU accelerated generalised subclass discriminant analysis for event and concept detection in video. In *Proc. ACM MM*. Brisbane, Australia, 1219–1222.

[2] S. Arestis-Chartampilas, N. Gkalelis, and V. Mezaris. 2016. AKSDA-MSVM: A GPU-accelerated multiclass learning framework for multimedia. In *Proc. ACM MM*. Amsterdam, The Netherlands, 461–465.

[3] G. Awad, C. G. M. Snoek, A. F. Smeaton, and Georges Quénot. 2016. TRECVid semantic indexing of video: A 6-year retrospective. *ITE Transactions on Media Technology and Applications* 4, 3 (2016), 187–208.

[4] G. Baudat and F. Anouar. 2000. Generalized discriminant analysis using a kernel approach. *Neural Comput.* 12, 10 (October 2000), 2385–2404.

[5] D. Cai, X. He, and J. Han. 2008. SRDA: An efficient algorithm for large-scale discriminant analysis. *IEEE Trans. on Knowl. and Data Eng.* 20, 1 (January 2008), 1–12.

[6] D. Cai, X. He, and J. Han. 2011. Speed up kernel discriminant analysis. *The VLDB Journal* 20, 1 (February 2011), 21–33.

[7] C.-C. Chang and C.-J. Lin. 2011. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2, 3 (2011), 27:1–27:27.

[8] X. Chang, Y. Yang, A. G. Hauptmann, E. P. Xing, et al. 2015. Semantic concept discovery for large-scale zero-shot event detection. In *Proc. Int. Conf. on Artificial Intelligence*. Buenos Aires, Argentina, 2234–2240.

[9] C. Chatterjee and V. P. Roychowdhury. 1997. On self-organizing algorithms and networks for class-separability features. *IEEE Trans. Neural Networks.* 8, 3 (May 1997), 663–678.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, et al. 2009. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE CVPR*. Miami, Florida, USA, 248–255.

[11] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, et al. 2014. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proc. ACM ICML*. Beijing, China, 647–655.

[12] A. Farhadi, I. Endres, D. Hoiem, and D. A. Forsyth. 2009. Describing objects by their attributes. In *Proc. IEEE CVPR*. Miami, Florida, USA, 1778–1785.

[13] K. Fukunaga. 1990. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA.

[14] D. Galanopoulos, F. Markatopoulou, V. Mezaris, and I. Patras. 2017. Concept Language Models and Event-based Concept Number Selection for Zero-example Event Detection. In *Proc. ACM ICMR*. Bucharest, Romania, 397–401.

[15] Y. A. Ghassabeh, F. Rudzicz, and H. A. Moghaddam. 2015. Fast incremental LDA feature extraction. *Pattern Recognit.* 48, 6 (June 2015), 1999–2012.

[16] N. Gkalelis and V. Mezaris. 2014. Video event detection using generalized subclass discriminant analysis and linear support vector machines. In *Proc. ACM ICMR*. Glasgow, UK, 25:25–25:32.

[17] N. Gkalelis and V. Mezaris. 2015. Accelerated nonlinear discriminant analysis. *arXiv preprint arXiv:1504.07000* (2015).

[18] N. Gkalelis and V. Mezaris. [n. d.]. Accelerated kernel discriminant analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* ([n. d.]), under review.

[19] G. H. Golub and C. F. van Loan. 2013. *Matrix computations* (4th ed.). The Johns Hopkins University Press, Baltimore, MD, USA.

[20] A. Habibian, K. E. A. van de Sande, and C. G. M. Snoek. 2013. Recommendations for video event recognition using concept vocabularies. In *Proc. ACM ICMR*. Dallas, Texas, USA, 89–96.

[21] T. Hofmann, B. Schölkopf, and A. J. Smola. 2008. Kernel methods in machine learning. *Annals of Statistics* 36, 3 (2008), 1171–1220.

[22] Z. Huang, H. T. Shen, J. Shao, S. Rüger, et al. 2008. Locality condensation: A new dimensionality reduction method for image retrieval. In *Proc. ACM MM*. Vancouver, British Columbia, Canada, 219–228.

[23] S. Ji and J. Ye. 2008. Generalized linear discriminant analysis: A unified framework and efficient model selection. *IEEE Trans. Neural Netw.* 19, 10 (October 2008), 1768–1782.

[24] T.-K. Kim, S.-F. Wong, B. Stenger, J. Kittler, et al. 2007. Incremental linear discriminant analysis using sufficient spanning set approximations. In *Proc. IEEE CVPR*. Minneapolis, Minnesota, USA.

[25] C. H. Lampert, H. Nickisch, and S. Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *Proc. IEEE CVPR*. 951–958.

[26] B. Leibe and B. Schiele. 2003. Analyzing appearance and contour based methods for object categorization. In *Proc. IEEE CVPR*, Vol. 2. Madison, WI, USA, 409–415.

[27] F.-F. Li, R. Fergus, and P. Perona. 2006. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* 65, 4 (April 2006), 594–611.

[28] L.-P. Liu, Y. Jiang, and Z.-H. Zhou. 2009. Least square incremental linear discriminant analysis. In *Proc. IEEE Int. Conf. on Data Mining*. Miami, Florida, USA, 298–306.

[29] F. Markatopoulou, A. Ioannidou, C. Tzelepis, T. Mironidis, et al. 2015. ITI-CERTH participation to TRECVID 2015. In *Proc. TRECVID*. Gaithersburg, MD, USA.

[30] J. Mercer. 1909. Functions of positive and negative type, and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc., London* 209 (1909), 415–446.

[31] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, et al. 2000. Invariant feature extraction and classification in kernel spaces. In *Proc. NIPS*. Denver, CO, USA, 526–532.

[32] H. A. Moghaddam and M. Matinfar. 2007. Fast adaptive LDA using quasi-Newton algorithm. *Pattern Recognit. Letters* 28, 5 (January 2007), 613–621.

[33] H. A. Moghaddam, M. Matinfar, S. M. S. Sadough, and K. A. Zadeh. 2005. Algorithms and networks for accelerated convergence of adaptive LDA. *Pattern Recognit.* 38, 4 (April 2005), 473–483.

[34] K.R. Muller, S. Mika, G. Ratsch, S. Tsuda, et al. 2001. An Introduction to Kernel-Based learning Algorithms. *IEEE Trans. Neural Netw.* 12, 2 (March 2001), 181–202.

[35] K. Ota, M. S. Dao, V. Mezaris, and F. G. B. De Natale. 2017. Deep learning for mobile multimedia: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.* 13, 3s (June 2017), 34:1–34:22.

[36] P. Over, G. Awad, J. G. Fiscus, B. Antonishek, et al. 2010. TRECVID 2010 - an overview of the goals, tasks, data, evaluation mechanisms, and metrics. In *Proc. TRECVID*. Gaithersburg, MD, USA.

[37] S. Pang, S. Ozawa, and N. Kasabov. 2005. Incremental linear discriminant analysis for classification of data streams. *IEEE Trans. Systems, Man, and Cybernetics, Part B* 35, 5 (September 2005), 905–914.

[38] C. H. Park and H. Park. 2005. Nonlinear discriminant analysis using kernel functions and the generalized singular value decomposition. *SIAM J. Matrix Analysis Applications* 27, 1 (2005), 87–102.

[39] Y. Peng, S. Pang, G. Chen, A. Sarrafzadeh, et al. 2013. Chunk incremental IDR/QR LDA learning. In *Proc. Int. Joint Conf. on Neural Networks*. Dallas, TX, USA, 1–8.

[40] N. Pittaras, F. Markatopoulou, V. Mezaris, and I. Patras. 2017. Comparison of fine-tuning and extension strategies for deep convolutional neural networks. In *Proc. Multimedia Modeling*, Vol. 10132. Reykjavik, Iceland, 102–114.

[41] S. Robertson. 2016. A new interpretation of average precision. In *Proc. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*. Singapore, Singapore, 689–690.

[42] V. Roth and V. Steinhage. 1999. Nonlinear discriminant analysis using kernel functions. In *Proc. NIPS*. Denver, Colorado, USA, 568–574.

[43] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. 2010. Adapting visual category models to new domains. In *Proc. ECCV*. Heraklion, Crete, Greece, 213–226.

[44] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, et al. 2015. Going deeper with convolutions. In *Proc. IEEE CVPR*. Boston, MA, USA, 1–9.

[45] T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars. 2015. A deeper look at dataset bias. In *Proc. German Conf. on Pattern Recognit.* Aachen, Germany, 504–516.

[46] T. Tommasi and T. Tuytelaars. 2014. A testbed for cross-dataset analysis. In *Proc. ECCV Workshops*. Zurich, Switzerland, 18–31.

[47] C. Tzelepis, D. Galanopoulos, V. Mezaris, and I. Patras. 2016. Learning to detect video events from zero or very few video examples. *Image Vision Comput.* 53 (2016), 35–44.

[48] C. Tzelepis, N. Gkalelis, V. Mezaris, and I. Kompatsiaris. 2013. Improving event detection using related videos and relevance degree support vector machines. In *Proc. ACM MM*. Barcelona, Spain, 673–676.

[49] C. Tzelepis, Z. Ma, V. Mezaris, B. Ionescu, et al. 2016. Event-based media processing and analysis: A survey of the literature. *Image Vision Comput.* 53 (2016), 3–19.

[50] V. Vapnik. 1998. *Statistical learning theory*. New York: Willey.

[51] H. Wang and C. Schmid. 2013. Action recognition with improved trajectories. In *Proc. IEEE ICCV*. Sydney, Australia, 3551–3558.

[52] Q. Wang and L. Zhang. 2012. Least squares online linear discriminant analysis. *Expert Syst. Appl.* 39, 1 (January 2012), 1510–1517.

[53] T. Xiao, J. Zhang, K. Yang, Y. Peng, et al. 2014. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proc. ACM MM*. Orlando, Florida, USA, 177–186.

[54] J. Ye. 2007. Least squares linear discriminant analysis. In *Proc. ACM ICML*. Corvalis, Oregon, USA, 1087–1093.

[55] J. Ye, Q. Li, H. Xiong, H. Park, et al. 2004. IDR/QR: An incremental dimension reduction algorithm via QR decomposition. In *Proc. ACM SIGKDD*. Seattle, Washington, USA, 364–373.

[56] H. Zhao and P. C. Yuen. 2008. Incremental linear discriminant analysis for face recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 38, 1 (January 2008), 210–221.