




Adaptive Classification Using Incremental Linearized Kernel Embedding

John Joseph Hall , *Member, IEEE*, Christopher Robbiano , *Member, IEEE*,
and Mahmood R. Azimi-Sadjadi , *Life Member, IEEE*

Abstract—This paper considers the problem of adaptive classification for performing pattern discrimination in varying conditions when new data arrives. A new efficient method is presented to incrementally update features in a Nyström-approximated linearized kernel embedding (LKE). Our method leverages a fast eigendecomposition algorithm for symmetric Arrowhead matrices. The proposed method can also be applied to kernel principal component analysis (KPCA) or similar problems. A mechanism is proposed which allows the incremental linearized kernel embedding to be used for updating of dictionaries in a sparse representation-based classification algorithm. The method is based on transporting the dictionaries into the embedding expanded with new data points and avoids the need to learn new dictionary matrices every time new data becomes available. The effectiveness of the developed methods is illustrated on two handwritten digit image data sets namely MNIST and USPS. Classification performance before and after sequential embedding updates is evaluated and compared. Comparisons are also made between our incremental LKE algorithm and the conventional approach to updating the empirical kernel map in terms of their computational requirements and numerical stability.

Index Terms—Kernel based lifelong learning, Kernel PCA, dictionary learning, Eigenvalue decomposition, adaptive classification, Kernel Dictionary Learning.

I. INTRODUCTION

THIS paper is concerned with adaptive classification using incremental parameter updating for potential lifelong learning in varying conditions. In particular, a sparse representation-based classification (SRC) framework which uses sparse coding and dictionary learning [1] in conjunction with a matched subspace classifier [2] and the linearized kernel embedding of [3] is considered here. The problem involves updating a previously trained dictionary, for use in a discriminative system, with information learned from new samples in a manner that improves classification performance for those newly learned

samples without sacrificing the performance of the system on the previously learned samples.

A number of methods currently exist for dictionary updating using information learned from new samples. The incremental K-SVD (IK-SVD) method [4] performs an incremental update to a previously learned dictionary. Dictionary atoms are updated one at a time while allowing for the addition of new atoms when needed. An entropy-based criterion is used to select the initial values of the new atoms by first sparsely coding the new data using the old dictionary matrix and then computing the entropy of each sparse coefficient vector. The samples with the largest entropy are used to initialize the new atoms. These samples correspond to the least sparse samples that cannot be accurately represented by the old dictionary matrix. The method of incremental dictionary update (IDU) in [5], [6] is based on an algorithm for joint incremental dictionary learning and sparse coding. A fast orthogonal matching pursuit (OMP) algorithm [7], [8] is used for sparse coding while allowing in-situ learning on unseen data to be carried out with substantially reduced computational needs when compared to IK-SVD [4].

Kernel machines are widely used for many complex pattern recognition and machine learning applications. However, most kernel machines require the formation of an exhaustive kernel matrix utilizing all training data samples. In [3], a linearized Kernel Dictionary Learning (LKDL) was proposed for reducing the complexity of representations. The LKDL method finds new lower-dimensional data representations based on the Nyström method. In this way, LKDL allows for linear dictionary learning methods to be applied to non-linearly mapped virtual features without increasing the dimensionality of the data representation. That is, discriminative benefits of kernel machines can be gained without the need for the full kernel matrix. The inherent sampling in this method is highly beneficial to developing adaptive classification systems with long-term incremental learning. Owing to all these benefits we adopted this method in this paper.

In the realm of machine vision and pattern recognition, there are countless problems which require a machine to continuously adapt over its lifetime to accommodate distribution changes of data samples coming from environments with varying conditions. Lifelong learning approaches seek to provide mechanisms for a machine to continually adapt to arriving data without inducing catastrophic forgetting or interference on examples

Manuscript received February 1, 2021; revised August 14, 2021, November 19, 2021, and February 24, 2022; accepted March 14, 2022. Date of publication March 31, 2022; date of current version April 28, 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jarvis Haupt. This work was supported by the Strategic Environmental Research and Development Program Contract under Grant W912HQ-17-C-0002. (Corresponding author: John Hall.)

The authors are with the Department of Electrical and Computer Engineering at Colorado State University, Fort Collins, CO 80523 USA (e-mail: halljj2@colostate.edu; cp.robbiano@gmail.com; azimi@engr.colostate.edu).

Digital Object Identifier 10.1109/TSP.2022.3162407

from previous environments. An example of such real scenarios is in underwater target classification [9]–[13] where the system faces the challenging task of identifying buried, or partially buried targets from sonar returns at several observation points. The targets can be found in a wide variety of environmental and operational conditions, making the need for model adaptation a concern of paramount importance.

To this end, this paper presents an enhancement to the theory of LKDL for Lifelong Learning. First, we present the background and geometric perspective of LKDL, and review how the method for attaining LKDL virtual samples proposed in [3] is an optimal transformation in the least-squares sense. Second, we discuss two incremental procedures for updating the linearized embedding when new important samples are made available. The first is the method of Hallgren and Northrop [14] which relies upon two rank-one eigenupdates. The second is our novel method, inspired by the work in [14], which we refer to as Symmetric Arrowhead Updating (SAU). This method uses Arrowhead eigendecompositions [15] for faster updating of the linearized kernel embedding. Complexity analysis of the method and a comparison with that of the method in [14] is also presented. The proposed method, like that of [14], can also be applied to standard kernel PCA [16]. Additionally, we present a method for transforming a dictionary learned on a previous embedding to the updated embedding and a strategy for incrementally updating the transformed dictionary. Our proposed method is tailored for use with dictionaries learned from LKDL virtual samples, and differs from IK-SVD [4] and IDU [5] in two ways: (a) the dimension of the dictionary atoms has the opportunity to increase, rather than simply increasing the number of atoms in the dictionary, as is done in IK-SVD and IDU, and (b) the proposed method provides a closed-form solution for transforming the dictionary to the updated embedding, and does not require any iterative optimization. An error analysis of this method is also provided. The proposed methods were then tested on two handwritten digit data sets, the MNIST and USPS data sets.

The remainder of the paper is organized as follows. In Section II, background on the Nyström method is presented along with the method for finding *virtual samples* from the set of *important samples*. The geometric interpretation of LKDL is also reviewed here. In Section III, we first review the method of Hallgren and Northrop for incrementally updating an embedding, and then introduce our novel SAU algorithm for accomplishing the same more efficiently. Section III-C presents a complexity analysis of our method along with a timing comparison of different methods for sequential eigendecomposition of a growing kernel matrix. Our proposed dictionary updating method, for rapidly updating K-SVD dictionaries, is proposed in Section IV. The error analysis of this update is also included here. Section IV briefly explains the decision procedure used in the modified Matched Subspace Classifier (MSC). Section V presents results of a classification experiment on the MNIST and USPS handwritten digit data sets. Lastly, Section VI gives some concluding remarks on the proposed methods.

II. LINEARIZED KERNEL EMBEDDING AND NYSTRÖM APPROXIMATION – A REVIEW

In order to allow for linear (dictionary or other) learning on kernelized representatives of data samples, a popular approach is to factorize the kernel matrix into a set of vector samples whose Gram matrix is approximately equal to the original kernel matrix [3], [17]. This can be achieved using a low-rank projection of the full mapped training data set via the Nyström approximation [3], [17], [18] which is briefly reviewed next.

A. Nyström Method for Kernel Matrix Approximation

Let $\mathbf{Z} = [\mathbf{z}_1 \mathbf{z}_2 \cdots \mathbf{z}_N]$ be a data matrix containing N data (feature) vectors $\mathbf{z}_i \in \mathbb{R}^d$ and $\Phi = [\phi(\mathbf{z}_1) \phi(\mathbf{z}_2) \cdots \phi(\mathbf{z}_N)] \in \mathbb{R}^{D \times N}$ be the corresponding nonlinearly mapped data matrix with columns $\phi(\mathbf{z}_i) \in \mathbb{R}^D$ where usually $D \gg d$. Then, the associated kernel Gram matrix of the nonlinearly mapped data is $\mathbf{K} = \Phi^\top \Phi$ where each element $\mathbf{K}_{i,j} = \kappa(\mathbf{z}_i, \mathbf{z}_j) = \phi(\mathbf{z}_i)^\top \phi(\mathbf{z}_j)$ and $\kappa(\mathbf{z}_i, \mathbf{z}_j)$ is an appropriate kernel function [19].

We assume that $c \leq N$ columns of \mathbf{K} are chosen according to a sampling scheme, e.g., uniform sampling or column-norm sampling [20]. The kernel matrix \mathbf{K} can be partitioned as,

$$\mathbf{K} = \begin{bmatrix} \mathbf{W} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{B} \end{bmatrix},$$

where \mathbf{W} is the kernel matrix associated with the retained *important* samples, $\mathbf{B} \in \mathbb{R}^{(N-c) \times (N-c)}$ is the one associated with unimportant samples, and \mathbf{A} is a matrix of inner products between the c chosen samples and $N - c$ unimportant samples. That is, if the data matrix associated with the retained samples is denoted by $\mathbf{Z}_R = [\mathbf{z}_1 \cdots \mathbf{z}_c] \in \mathbb{R}^{d \times c}$ and its mapped version in the feature space by $\Phi_R = [\phi(\mathbf{z}_1) \cdots \phi(\mathbf{z}_c)] \in \mathbb{R}^{D \times c}$, then $\mathbf{W} = \Phi_R^\top \Phi_R \in \mathbb{R}^{c \times c}$.

Now, denoting $\mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{A} \end{bmatrix} \in \mathbb{R}^{N \times c}$ and using the Nyström method [3], [17], [18], the kernel matrix \mathbf{K} can be approximated as

$$\begin{aligned} \mathbf{K} &\approx \hat{\mathbf{K}} = \mathbf{C} \mathbf{W}^{-1} \mathbf{C}^\top \\ &= \Phi^\top \Phi_R (\Phi_R^\top \Phi_R)^{-1} \Phi_R^\top \Phi \\ &= \Phi^\top \mathbf{P}_{\Phi_R} \Phi, \end{aligned} \quad (1)$$

where \mathbf{W} is assumed to be full rank and \mathbf{P}_{Φ_R} denotes the orthogonal projection matrix onto the span of Φ_R . Since \mathbf{P}_{Φ_R} is idempotent, (1) implies that $\hat{\mathbf{K}}$ is the Gram matrix after projecting all samples in Φ onto the subspace $\langle \Phi_R \rangle$.

B. Linearized Kernel Embedding

The method presented in [3] finds a linear realization of the kernel matrix with features that capture the benefits of the nonlinear mapping in an efficient way. More specifically, using this method the kernel Gram matrix \mathbf{K} can be approximately

factored into $\mathbf{K} \approx \mathbf{F}^\top \mathbf{F}$ leading to the *virtual feature* matrix \mathbf{F} that yields an approximation of the full kernel matrix.

To see this, let us apply the eigenvalue decomposition to the symmetric full rank kernel matrix \mathbf{W} such that $\mathbf{W} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^\top$ with $\mathbf{\Sigma}$ being a diagonal matrix containing all the eigenvalues and \mathbf{V} is an orthogonal matrix containing the associated eigenvectors as its columns. Using this eigenvalue decomposition, (1) can be expressed as,

$$\mathbf{K} \approx \mathbf{C}\mathbf{W}^{-1}\mathbf{C}^\top = \mathbf{F}^\top \mathbf{F} = \mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{V}^\top \mathbf{C}^\top,$$

This yields the virtual data matrix \mathbf{F} as,

$$\mathbf{F} = \mathbf{\Sigma}^{-1/2} \mathbf{V}^\top \mathbf{C}^\top. \quad (2)$$

Thus, for a particular data sample \mathbf{z} and its mapped version $\phi(\mathbf{z})$, the corresponding virtual feature vector is,

$$\mathbf{f} = \mathbf{\Sigma}^{-1/2} \mathbf{V}^\top \mathbf{\Phi}_R^\top \phi(\mathbf{z}) \quad (3)$$

$$= \mathbf{\Sigma}^{-1/2} \mathbf{V}^\top [\kappa(\mathbf{z}_1, \mathbf{z}) \cdots \kappa(\mathbf{z}_c, \mathbf{z})]^\top. \quad (4)$$

The dimension of the virtual samples in \mathbf{F} can be reduced by choosing the k largest eigenvalues, $\mathbf{\Sigma}_k = \text{diag}[\sigma_1 \sigma_2 \cdots \sigma_k]$ of \mathbf{W} and the associated eigenvectors $\mathbf{V}_k = [\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_k]$. Then, the reduced dimensional *virtual data matrix* [3] is given by

$$\mathbf{F}_k = \mathbf{\Sigma}_k^{-1/2} \mathbf{V}_k^\top \mathbf{C}^\top. \quad (5)$$

Given the matrices $\mathbf{\Sigma}_k^{-1/2}$ and \mathbf{V}_k , a particular data sample \mathbf{z} can be mapped to the corresponding virtual feature vector using,

$$\mathbf{f} = \mathbf{\Sigma}_k^{-1/2} \mathbf{V}_k^\top [\kappa(\mathbf{z}_1, \mathbf{z}) \cdots \kappa(\mathbf{z}_c, \mathbf{z})]^\top.$$

where $\mathbf{z}_1 \cdots \mathbf{z}_c$ are the c important samples, chosen via sampling, and $\kappa(\mathbf{x}, \mathbf{y})$ is the chosen kernel function.

C. Geometric Perspective of Kernel Embedding

At first glance, one might assume that the approximation $\hat{\mathbf{K}} = \mathbf{F}^\top \mathbf{F}$ gives a solution that is unrelated to the original embedded signals and their geometry. As explained above, the Nyström method can be derived by projecting the full data set onto the subspace defined by the important samples [21]. In this section, we illustrate how the coordinates \mathbf{F} are in fact the exact local coordinates (i.e., using c -dimensional coordinates rather than D) of *all* samples in the feature space when projected onto the principal axes of the important samples, i.e., the $c \leq N$ samples selected for approximating the kernel matrix, in the feature space.

Proposition 2.1: Given a data sample \mathbf{z} , the principal component vector of the mapped data, $\phi(\mathbf{z})$, is its associated virtual feature vector \mathbf{f} , i.e., $\mathbf{f} = \mathbf{U}^\top \phi(\mathbf{z})$ where \mathbf{U}^\top is the mapping matrix.

Proof: The important samples in the kernel feature space, $\mathbf{\Phi}_R$, can be decomposed using the SVD as

$$\mathbf{\Phi}_R = \mathbf{U}\mathbf{\Sigma}^{1/2}\mathbf{V}^\top \in \mathbb{R}^{D \times c} \quad (6)$$

where $\mathbf{U} \in \mathbb{R}^{D \times c}$ is a semi-orthogonal matrix that contains the orthonormal left singular vectors of $\mathbf{\Phi}_R$, i.e., $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_{c \times c}$,

$\mathbf{\Sigma}^{1/2}$ is a $c \times c$ matrix of singular values, and \mathbf{V} is a $c \times c$ orthonormal matrix containing the right singular vectors.

Now, using (6) it is evident that the matrix \mathbf{U} diagonalizes the rank- c correlation matrix, $\mathbf{\Phi}_R \mathbf{\Phi}_R^\top$. Hence, the matrix \mathbf{U}^\top maps the nonlinearly mapped data $\phi(\mathbf{z})$ to the corresponding principal components. Furthermore, $\mathbf{f} = \mathbf{U}^\top \phi(\mathbf{z})$ can easily be confirmed by replacing $\mathbf{\Phi}_R$ in (3) with its SVD. \square

Remark 2.1: For any pair of data samples $\mathbf{z}_i, \mathbf{z}_j \in \mathbf{Z}_R$, using (3) and (6), we get $\mathbf{f}_i^\top \mathbf{f}_j = \phi(\mathbf{z}_i)^\top \mathbf{\Phi}_R \mathbf{V} \mathbf{\Sigma}^{-1/2} \mathbf{\Sigma}^{-1/2} \mathbf{V}^\top \mathbf{\Phi}_R^\top \phi(\mathbf{z}_j) = \phi(\mathbf{z}_i)^\top \mathbf{U} \mathbf{U}^\top \phi(\mathbf{z}_j)$. However, we also know that $\mathbf{P}_{\mathbf{\Phi}_R} = \mathbf{U} \mathbf{U}^\top$, i.e., the projection matrix onto the subspace spanned by \mathbf{U} , and also since $\phi(\mathbf{z}_i)$ is in $\langle \mathbf{\Phi}_R \rangle$ then $\mathbf{P}_{\mathbf{\Phi}_R} \phi(\mathbf{z}_i) = \phi(\mathbf{z}_i)$. Thus, $\mathbf{f}_i^\top \mathbf{f}_j = \phi(\mathbf{z}_i)^\top \phi(\mathbf{z}_j)$.

Furthermore, we have $\|\phi(\mathbf{z}_i) - \phi(\mathbf{z}_j)\|_2 = \|\mathbf{f}_i - \mathbf{f}_j\|_2 \forall i, j \in \{1, \dots, c\}$, in other words, this embedding preserves distance and angle properties, and \mathbf{f}_i simply gives a c -dimensional subspace coordinates representation for $\phi(\mathbf{z}_i) \in \langle \mathbf{\Phi}_R \rangle$. Similarly, if $k < c$ then \mathbf{f}_i 's represent the coordinates of the samples projected onto the first k principal axes of the important samples in $\mathbf{\Phi}_R$. This means that representations $\phi(\mathbf{z}_i)$ and \mathbf{f}_i are isometric for important samples.

III. UPDATING EMBEDDING WITH NEW IMPORTANT SAMPLES

In this section, we consider a scenario where the system trained with the original c samples needs to be updated when a new important sample $\mathbf{z}_{\text{new}} = \mathbf{z}_{c+1}$ is added. That is, the augmented data matrix of the chosen samples and its nonlinearly mapped version become $\tilde{\mathbf{Z}}_R = [\mathbf{Z}_R, \mathbf{z}_{c+1}] \in \mathbb{R}^{d \times (c+1)}$ and $\tilde{\mathbf{\Phi}}_R = [\phi(\mathbf{z}_1) \cdots \phi(\mathbf{z}_c) \phi(\mathbf{z}_{c+1})] \in \mathbb{R}^{D \times (c+1)}$, respectively. It should be noted that the procedures presented here are not limited to an important samples (Nyström approximation) approach, but can be used for standard kernel PCA as well.

Now, we desire to find a new set of virtual samples, $\tilde{\mathbf{F}}$, that satisfies

$$\tilde{\mathbf{F}} = \tilde{\mathbf{\Sigma}}^{-1/2} \tilde{\mathbf{V}}^\top \tilde{\mathbf{C}}^\top, \quad (7)$$

where $\tilde{\mathbf{\Sigma}}$ and $\tilde{\mathbf{V}}$ are traditionally computed by applying the eigenvalue decomposition (EVD) to the corresponding kernel matrix $\tilde{\mathbf{W}} = \tilde{\mathbf{\Phi}}_R^\top \tilde{\mathbf{\Phi}}_R$. More specifically,

$$\tilde{\mathbf{W}} = \tilde{\mathbf{V}} \tilde{\mathbf{\Sigma}} \tilde{\mathbf{V}}^\top = \begin{bmatrix} \mathbf{W} & \boldsymbol{\rho} \\ \boldsymbol{\rho}^\top & \kappa_{c+1, c+1} \end{bmatrix}, \quad (8)$$

where $\boldsymbol{\rho} = \mathbf{k}(\mathbf{Z}_R, \mathbf{z}_{c+1}) = [\kappa_{1, c+1} \kappa_{2, c+1} \cdots \kappa_{c, c+1}]^\top$ with $\kappa_{i, j} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$. However, here the goal is to accomplish this without resorting to any computationally demanding EVD algorithm.

Also, $\tilde{\mathbf{C}}$ in (7) is given by $\tilde{\mathbf{C}} = [\tilde{\mathbf{W}}^\top, \tilde{\mathbf{A}}^\top]^\top \in \mathbb{R}^{(N+1) \times (c+1)}$, where $\tilde{\mathbf{A}} = [\mathbf{A}, \boldsymbol{\zeta}] \in \mathbb{R}^{(N-c) \times (c+1)}$, with $\boldsymbol{\zeta} = \mathbf{k}(\mathbf{Z}_U, \mathbf{z}_{c+1}) = [\kappa(\mathbf{z}_{U,1}, \mathbf{z}_{c+1}) \cdots \kappa(\mathbf{z}_{U, N-c}, \mathbf{z}_{c+1})]^\top$ and $\mathbf{z}_{U, i} \in \mathbf{Z}_U = \mathbf{Z} \setminus \mathbf{Z}_R$, i.e., sample set containing $N - c$ unimportant samples discarded as a result of the sampling.

The first approach for the incremental embedding update presented in this section uses the information about the current

Algorithm 1: Incremental Eigendecomposition of Kernel Matrix [14].

Require: The augmented data matrix $\tilde{\mathbf{Z}}_R$, matrices \mathbf{V} and Σ for kernel matrix \mathbf{W} and the kernel function $\kappa(\cdot, \cdot)$.

Ensure: Eigenvalue and eigenvector matrices $\tilde{\Sigma}$ and $\tilde{\mathbf{V}}$ of $\tilde{\mathbf{W}}$

- 1: $\Sigma^0 \leftarrow \begin{bmatrix} \Sigma & \mathbf{0} \\ \mathbf{0}^\top & \kappa_{c+1,c+1}/4 \end{bmatrix}$
 - 2: $\mathbf{V}^0 \leftarrow \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0}^\top & \mathbf{1} \end{bmatrix}$
 - 3: $\sigma \leftarrow 4/\kappa_{c+1,c+1}$
 - 4: $\boldsymbol{\nu}_1 \leftarrow [\kappa_{1,c+1}, \kappa_{2,c+1}, \dots, \kappa_{c+1,c+1}/2]^\top$
 - 5: $\boldsymbol{\nu}_2 \leftarrow [\kappa_{1,c+1}, \kappa_{2,c+1}, \dots, \kappa_{c+1,c+1}/4]^\top$
 - 6: $\Sigma^1, \mathbf{V}^1 \leftarrow \text{rankoneupdate}(\sigma, \boldsymbol{\nu}_1, \Sigma^0, \mathbf{V}^0)$
 - 7: $\tilde{\Sigma}, \tilde{\mathbf{V}} \leftarrow \text{rankoneupdate}(-\sigma, \boldsymbol{\nu}_2, \Sigma^1, \mathbf{V}^1)$
-

\mathbf{W} matrix, and its eigenpairs, to compute the new eigenpairs of $\tilde{\mathbf{W}}$. This is accomplished by utilizing the rank-one update procedure in [14], [22].

Let us define $\boldsymbol{\nu}_1 = [\boldsymbol{\rho}^\top \frac{1}{2} \kappa_{c+1,c+1}]^\top$, $\boldsymbol{\nu}_2 = [\boldsymbol{\rho}^\top \frac{1}{4} \kappa_{c+1,c+1}]^\top$, and $\sigma = 4/\kappa_{c+1,c+1}$. Then, it can be shown [14] that $\tilde{\mathbf{W}}$ can be expressed as

$$\begin{aligned} \tilde{\mathbf{W}} &= \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0}^\top & \frac{1}{4} \kappa_{c+1,c+1} \end{bmatrix} + \sigma \boldsymbol{\nu}_1 \boldsymbol{\nu}_1^\top - \sigma \boldsymbol{\nu}_2 \boldsymbol{\nu}_2^\top, \\ &= \mathbf{W}^0 + \sigma \boldsymbol{\nu}_1 \boldsymbol{\nu}_1^\top - \sigma \boldsymbol{\nu}_2 \boldsymbol{\nu}_2^\top \end{aligned} \quad (9)$$

corresponding to a diagonal expansion of \mathbf{W} and two rank one updates, where $\mathbf{0}$ is a column vector of zeros of appropriate dimension. As can be seen from (9) the matrix \mathbf{W}^0 has an additional eigenvalue $\frac{1}{4} \kappa_{c+1,c+1}$ and corresponding eigenvector $\mathbf{v}_{c+1}^0 = [0, \dots, 1]^\top$ when compared to the eigendecomposition of \mathbf{W} . All other eigenvalues are the same as those of \mathbf{W} with corresponding eigenvectors $\mathbf{v}_i^0 = [\mathbf{v}_i^\top, 0]^\top$, $i \in \{1, \dots, c\}$. Matrix \mathbf{W}^0 is symmetric positive semi-definite and it will remain symmetric positive semi-definite after the addition and subtraction of two rank one Gram matrices $\sigma \boldsymbol{\nu}_1 \boldsymbol{\nu}_1^\top$ and $\sigma \boldsymbol{\nu}_2 \boldsymbol{\nu}_2^\top$, respectively.

The steps in the algorithm for incrementally calculating $\tilde{\mathbf{W}}$ from \mathbf{W} are listed in the following table. Note that \mathbf{V}^1 and Σ^1 in step 6 correspond to the intermediate eigendecomposition of $\mathbf{W}^1 := \mathbf{W}^0 + \sigma \boldsymbol{\nu}_1 \boldsymbol{\nu}_1^\top$ after the first rank one update whereas the results of step 7 are the final desired $\tilde{\mathbf{V}}$ and $\tilde{\Sigma}$.

In the following two subsections we first review the method for incrementally solving the rank-one update of a symmetric matrix given a known perturbation required for the last two steps in Algorithm 1. We then propose a new and more efficient approach to the two rank-one updates method.

A. Rank One Perturbation Eigendecomposition Update

Given the eigendecomposition of the symmetric matrix $\mathbf{W}^0 = \mathbf{V}^0 \Sigma^0 \mathbf{V}^{0\top}$ where \mathbf{V}^0 is an orthogonal matrix, we define

the *perturbed* matrix

$$\begin{aligned} \mathbf{B} &= \mathbf{V}^0 \Sigma^0 \mathbf{V}^{0\top} + \sigma \boldsymbol{\nu} \boldsymbol{\nu}^\top \\ &= \mathbf{V}^0 (\Sigma^0 + \sigma \boldsymbol{\tau} \boldsymbol{\tau}^\top) \mathbf{V}^{0\top} \end{aligned}$$

where $\boldsymbol{\tau} = \mathbf{V}^{0\top} \boldsymbol{\nu}$. Using the eigendecomposition of $\Sigma^0 + \sigma \boldsymbol{\tau} \boldsymbol{\tau}^\top = \mathbf{U} \Sigma^1 \mathbf{U}^\top$ [23], the eigendecomposition of \mathbf{B} can be found by $\mathbf{B} = \mathbf{V}^1 \Sigma^1 \mathbf{V}^{1\top}$ where $\mathbf{V}^1 := \mathbf{V}^0 \mathbf{U}$. Since both \mathbf{V}^0 and \mathbf{U} are orthogonal matrices, \mathbf{V}^1 will also be an orthogonal matrix. The eigenvalues of \mathbf{B} can be computed by finding the roots of the secular equation [24]. The eigenvectors of the perturbed matrix \mathbf{B} are given by [23]

$$\mathbf{v}_i^1 = \frac{\mathbf{V}^0 \mathbf{D}_i^{-1} \boldsymbol{\tau}}{\|\mathbf{D}_i^{-1} \boldsymbol{\tau}\|} \quad (10)$$

where $\mathbf{D}_i := \Sigma^0 - \lambda_i' \mathbf{I}$ where λ_i' is the i^{th} eigenvalue of \mathbf{B} . This method exhibits some numerical stability issues for invertible, but poorly conditioned matrices. Alternatively, a stabilized approach, e.g., the one in [22], can be utilized for applications where many rank one iterations are necessary.

B. An Alternate Approach Using Arrowhead Updates

In this section we present a novel approach to solving the expanding eigendecomposition problem using arrowhead matrix eigendecompositions [15]. In this approach, the update from the eigendecomposition of \mathbf{W} to that of $\tilde{\mathbf{W}}$ can be done by noting that the addition made to \mathbf{W}^0 to form $\tilde{\mathbf{W}}$ yields an addition to the eigenvalue matrix of \mathbf{W}^0 resulting in an arrowhead matrix. This arrowhead matrix's eigendecomposition can be obtained more efficiently compared to the method in the previous section which uses two consecutive rank-one updates. The eigendecomposition of this arrowhead matrix is directly related to the eigendecomposition of $\tilde{\mathbf{W}}$ through \mathbf{W}^0 's eigenvectors matrix.

As before, we represent $\mathbf{W} = \mathbf{V} \Sigma \mathbf{V}^\top$ and $\tilde{\mathbf{W}} = \mathbf{W}^0 + \Gamma$, where \mathbf{W}^0 was defined before and

$$\Gamma = \begin{bmatrix} \mathbf{0} & \boldsymbol{\rho} \\ \boldsymbol{\rho}^\top & \frac{3}{4} \kappa_{c+1,c+1} \end{bmatrix}$$

We also noted that $\mathbf{W}^0 = \mathbf{V}^0 \Sigma^0 \mathbf{V}^{0\top}$ where

$$\mathbf{V}^0 = \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0}^\top & \mathbf{1} \end{bmatrix}$$

and

$$\Sigma^0 = \begin{bmatrix} \Sigma & \mathbf{0} \\ \mathbf{0}^\top & \frac{1}{4} \kappa_{c+1,c+1} \end{bmatrix}$$

Now, it is easy to see that $\mathbf{V}^{0\top} \Gamma \mathbf{V}^0$ has the following form

$$\mathbf{V}^{0\top} \Gamma \mathbf{V}^0 = \begin{bmatrix} \mathbf{0} & \mathbf{V}^\top \boldsymbol{\rho} \\ \boldsymbol{\rho}^\top \mathbf{V} & \frac{3}{4} \kappa_{c+1,c+1} \end{bmatrix}$$

Note that this is nearly an arrowhead matrix, all it is missing is the diagonal. Therefore, using a similar approach to a typical rank one perturbation problem, we can factor $\tilde{\mathbf{W}}$ as

$$\tilde{\mathbf{W}} = \mathbf{W}^0 + \Gamma$$

TABLE I
COMPLEXITY COMPARISON (LEADING TERMS)

Hallgren et. al.	Time Complexity	Memory Complexity
Step 6	$2c^3 + \mathcal{O}(c^2)$	$\mathcal{O}(c^2)$
Step 7	$2c^3 + \mathcal{O}(c^2)$	$\mathcal{O}(c^2)$
Total	$4c^3 + \mathcal{O}(c^2)$	$\mathcal{O}(c^2)$
SAU Eig. (Ours)	Time Complexity	Memory Complexity
Step 5	$\mathcal{O}(c^2)$	$\mathcal{O}(c^2)$
Step 6	$2c^3$	$\mathcal{O}(c^2)$
Total	$2c^3 + \mathcal{O}(c^2)$	$\mathcal{O}(c^2)$

Algorithm 2: SAU Eigendecomposition of Growing Kernel Matrix.

Require: Matrices \mathbf{V} , Σ , vector ρ , and scalar $\kappa_{c+1,c+1}$.

Ensure: Eigenvalue and eigenvector matrices $\tilde{\Sigma}$ and $\tilde{\mathbf{V}}$ of $\tilde{\mathbf{W}}$

- 1: $\Sigma^0 \leftarrow \begin{bmatrix} \Sigma & \mathbf{0} \\ \mathbf{0}^\top & \kappa_{c+1,c+1}/4 \end{bmatrix}$
- 2: $\mathbf{V}^0 \leftarrow \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
- 3: $\pi \leftarrow \mathbf{V}^\top \rho$
- 4: $\Gamma \leftarrow \begin{bmatrix} \mathbf{0} & \pi \\ \pi^\top & \frac{3}{4}\kappa_{c+1,c+1} \end{bmatrix}$
- 5: $\mathbf{U}, \tilde{\Sigma} \leftarrow \text{aheig}(\Sigma^0 + \Gamma)$ [15]
- 6: $\tilde{\mathbf{V}} \leftarrow \mathbf{V}^0 \mathbf{U}$

$$\begin{aligned}
 &= \mathbf{V}^0 \Sigma^0 \mathbf{V}^{0\top} + \Gamma \\
 &= \mathbf{V}^0 (\Sigma^0 + \mathbf{V}^{0\top} \Gamma \mathbf{V}^0) \mathbf{V}^{0\top} \quad (11)
 \end{aligned}$$

The factor in parentheses in (11) is an arrowhead matrix. Thus, using the method in [15], we can solve the eigendecomposition $\Sigma^0 + \mathbf{V}^{0\top} \Gamma \mathbf{V}^0 = \mathbf{U} \tilde{\Sigma} \mathbf{U}^\top$. Then, this eigendecomposition can be used to yield $\tilde{\mathbf{W}} = \mathbf{V}^0 (\mathbf{U} \tilde{\Sigma} \mathbf{U}^\top) \mathbf{V}^{0\top}$ which gives the eigenvector matrix $\tilde{\mathbf{V}} = \mathbf{V}^0 \mathbf{U}$. The steps in this algorithm are outlined below in Algorithm 2. It should be noted that this symmetric arrowhead update (SAU) procedure can be used on any growing real symmetric full rank matrix \mathbf{W} and is not limited to Gram matrices. Step 5 of this algorithm utilizes the arrowhead eigenvalue (aheig) decomposition (Algorithm 5) presented in [15].

C. Computational Complexity Analysis

A major motivation for the creation of the SAU updates method is the desire to reduce the computational cost of repeated model updates when varying data is arriving continuously. The proposed SAU algorithm provides a competitive approach to solving the incremental kernel PCA [19] and incremental empirical kernel map problems. A comparison of the computational cost of the SAU algorithm with that of [14] is presented in Table I which breaks down the leading terms for time and memory complexity for both methods.

As far as the SAU algorithm is concerned, step 5 relies on the arrowhead eigendecomposition of [15] which has $\mathcal{O}(c^2)$ time complexity and $\mathcal{O}(c)$ memory complexity. Step 6 is computationally the most expensive step of our procedure which involves

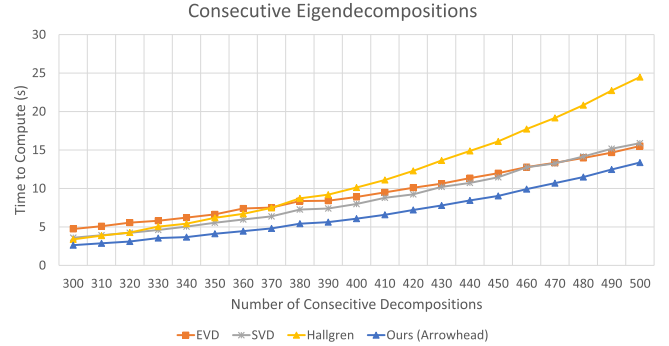


Fig. 1. Runtime Comparison for R consecutive Eigendecomposition.

matrix multiplication of two $c \times c$ matrices hence requiring approximately $2c^3$ operations. This cubic factor is difficult, if not impossible, to reduce, since the update procedure will always require rotating c eigenvectors. This brings our time complexity to $2c^3 + \mathcal{O}(c^2)$ and the memory complexity to $3c^2 + \mathcal{O}(c)$ due to the allocations for \mathbf{V}^0 , \mathbf{U} , and $\tilde{\mathbf{V}}$, respectively. In contrast, the method in [14] requires $4c^3 + \mathcal{O}(c^2)$ flops and also admits $\mathcal{O}(c^2)$ memory complexity. Therefore, our algorithm, like that of [14], admits $\mathcal{O}(c^3)$ time complexity and $\mathcal{O}(c^2)$ memory complexity but is approximately twice as efficient in the cubic factor. The experimental results in the next section attest to this fact.

D. Computational Time and Consistency Comparison

Here, we compare the computational requirements to carry out several consecutive eigendecompositions of a growing kernel matrix \mathbf{W} using the methods in Sections III A and B. More specifically, an experiment was conducted to determine which approach provides less computational time to perform a series of eigendecomposition of an expanding matrix. In this set up, all methods start with 1 sample and the corresponding Gram matrix \mathbf{W} matrix is created. This matrix is then expanded by a single sample at a time, R times.

Fig. 1 displays the time to fully compute all R for $R \in [300, 310, \dots, 490, 500]$ eigendecompositions via consecutive EVD (red plot) [25], consecutive SVD (gray) [25], two Rank-One Updates (orange) [14], and the proposed symmetric arrowhead update (blue) in Section III B. As evident from this figure, for the task of incrementally updating the eigendecomposition, the SAU approach provides the quickest solutions by harnessing the information of the previous eigendecomposition steps.

Lastly, the consistency of eigenvalues obtained via the SAU updates and the standard batch SVD of the important samples are studied. Fig. 2 gives the plot of the Frobenius norm of the difference between $\Sigma_i = \mathbf{S}_i$, the (diagonal) eigenvalues matrix computed sequentially via the incremental method in Algorithm 2, and $\Sigma_{svd} = \mathbf{S}_{svd}$, the eigenvalues matrix computed from decomposing all c samples simultaneously via the batch SVD. As can be seen, even after adding hundreds of important samples, the eigenvalues computed via our incremental approach are nearly identical to those obtained by performing the more expensive batch SVD, with the differences being on the order of 10^{-13} .

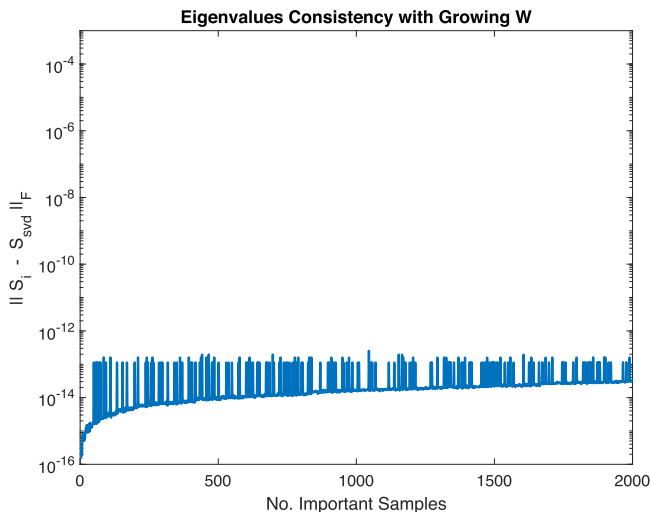


Fig. 2. Incremental vs. Batch SVD eigenvalues for growing \mathbf{W} .

IV. SRC MODEL AND DICTIONARY MATRIX UPDATING

In this section, we introduce a method for updating learned dictionaries of an SRC-based classifier [11] with information present in new samples. To start, as before let \mathbf{Z} be the entire sample set and $\mathbf{Z}_R \subset \mathbf{Z}$ be the important sample set selected via a particular sampling method. Each sample $\mathbf{z} \in \mathbf{Z}$ has an associated virtual sample representation $\mathbf{f} \in \mathbf{F}$. Also, let \mathbf{z}_{c+1} be a newly identified important sample. The augmented data matrix $\tilde{\mathbf{Z}}_R = [\mathbf{Z}_R, \mathbf{z}_{c+1}]$ has a corresponding new virtual data matrix denoted by $\tilde{\mathbf{F}}$. In what follows, we first discuss the Matched Subspace Classifier (MSC) [2] and its modified version. We then present a new mapping scheme for updating the dictionary matrices once new samples are to be added from new environments.

A. Modified MSC and SRC Framework

Let us consider a general M -ary classification problem in which the observations can come from $m = 1, \dots, M$ possible classes. The MSC [2] assumes a signal model of the form,

$$\mathbf{Z} = \mathbf{H}_m \Theta_m + \mathbf{N} \quad m \in [1, M] \quad (12)$$

where \mathbf{Z} is the data matrix containing observation vectors \mathbf{z}_i as its column, \mathbf{H}_m is the dictionary matrix whose columns are the basis vectors that span the subspace associated with the m^{th} class, Θ_m is an unknown matrix with the columns being the parameter vector associated with data vectors \mathbf{z}_i , and \mathbf{N} denotes an additive zero-mean noise matrix.

The decision-making in MSC is carried out by determining the class that satisfies

$$m^* = \operatorname{argmin}_{m \in [1, M]} \{ \|\mathbf{Z} - \mathbf{H}_m \hat{\Theta}_m\|_F^2 \} \quad (13)$$

where $\|\mathbf{A}\|_F^2 = \operatorname{tr}(\mathbf{A}\mathbf{A}^\top)$ represents squared Frobenius norm of the matrix \mathbf{A} and $\hat{\Theta}_m$ is the least squares (LS) estimate of Θ_m when dictionary matrix \mathbf{H}_m is used [2]. That is, the correct class m^* is the one which makes the magnitude of the reconstruction error the smallest. Training of the MSC amounts to constructing

class-dependent dictionary matrices $\mathbf{H}_m, m = 1, \dots, M$ from representative training data sets in each class m .

In this section, we employ the K-SVD dictionary learning method [1] to find \mathbf{H}_m matrices in (13) hence making the MSC a sparse representation classifier (SRC). Additionally, the virtual feature matrix \mathbf{F} is used instead of the actual observation matrix \mathbf{Z} to take full benefit of the nonlinear kernel mapping without increasing the dimension of the representation. The MSC incorporating these changes is referred to as “modified MSC” in the sequel. The incremental dictionary learning/updating method in this paper allows the previously trained modified MSC to incorporate the relevant knowledge embedded in the new data samples when faced with new conditions. More specifically, learning via evolving task-specific dictionaries allows the classification system to generalize previous training and hence perform well on both old and new tasks [26]–[30]. Moreover, the inherent sampling in the LKDL method covered in Section II significantly reduces the number of useful samples required to generate highly discriminative kernelized features.

In the modified MSC method [11], the decision-making rule becomes,

$$m^* = \operatorname{argmin}_{m \in [1, M]} \{ \|\mathbf{F} - \mathbf{H}_m \hat{\Theta}_m\|_F^2 \} \quad (14)$$

where $\hat{\Theta}_m$ is generated using a pursuit method such as the Orthogonal Matching Pursuit (OMP) or Basis Pursuit (BP) algorithms [31] when the dictionary \mathbf{H}_m is used. Here, we will use the fast OMP method in [8] which does not require any matrix inversion.

B. Joint Embedding and Dictionary Updating

Using the method presented in Section III-B, we know how to expand the eigendecomposition of a growing kernel matrix. This expanded eigendecomposition can be used to transfer virtual features from an old embedding to the new one before using the modified MSC. That is, the updated virtual features $\tilde{\mathbf{F}}$ are now used in the signal model hence providing important sample information from the new environment.

Nevertheless, the dictionary matrices \mathbf{H}_m that were generated from the original virtual feature vectors in \mathbf{F} may no longer suitably represent the new samples in the augmented data set $\tilde{\mathbf{Z}} = [\mathbf{Z}, \mathbf{z}_{\text{new}}]$. Clearly, retraining the dictionary matrices every time a new important sample is incorporated using e.g. K-SVD learning [1] becomes inefficient and impractical. Moreover, the IDU method [5] cannot be applied either since it only allows for the augmentation in the column space of the dictionary matrices and not the row space which occurs as a result of adding new important samples. Note that though each virtual sample $\mathbf{f} \in \mathbf{F}$ is associated with a sample $\mathbf{z} \in \mathbf{Z}$, it is a non-trivial task to find a representation of dictionary atoms $\mathbf{h} \in \mathbf{H}$ from the original sample subspace [32], [33]. Owing to these reasons, here we propose a mechanism to find a mapped dictionary in the new embedding that does not degrade the previous learning by using a coordinate transformation matrix \mathbf{T} . Compared to the repeated K-SVD application, this method of transforming old dictionaries to produce new class-dependent dictionaries for the new embedding is computationally more efficient. Moreover,

it allows the system designer to produce usable dictionaries in the new embedding *without* being required to carry along all the prior training data sets that would be needed for a full K-SVD retraining. Clearly, this would defeat the whole purpose of incremental learning particularly for lifelong learning applications. The computational complexities are discussed later in this section.

The motivation behind this mapping is that, since the old atoms can be completely represented as linear combinations in $\text{span}\{\mathbf{F}\}$, we ought to transform the atoms via the linear transformation which best matches (in the squared error sense) old \mathbf{f} 's to their new virtual-sample representatives in the updated embedding. For the sake of efficiency, we instead generate the mapping which optimally brings the virtual features of the important samples, referred to as \mathbf{F}_R , to their new representations.

Now, let $\mathbf{F}_R = \Sigma^{-1/2} \mathbf{V}^\top \mathbf{W}_+ \in \mathbb{R}^{c \times (c+b)}$ be the transformed features corresponding to *all* important samples, including the b new ones that were used in updating the embedding, from the set $\tilde{\mathbf{Z}}_R = [\mathbf{Z}_R, \mathbf{Z}_{\text{new}}] \in \mathbb{R}^{d \times (c+b)}$ in the *old* embedding. The original inner products matrix \mathbf{W} is augmented with the inner products matrix between old and new important samples to produce the \mathbf{W} -augmented matrix,

$$\mathbf{W}_+ = \begin{bmatrix} \mathbf{W} & \mathbf{C}_{\text{new}}^\top \end{bmatrix} \in \mathbb{R}^{c \times (c+b)},$$

with

$$\mathbf{C}_{\text{new}}^\top = \mathbf{k}(\mathbf{Z}_R, \mathbf{Z}_{\text{new}}) = \begin{bmatrix} \mathbf{k}(\mathbf{Z}_R, \mathbf{z}_{\text{new},1}) & \cdots & \mathbf{k}(\mathbf{Z}_R, \mathbf{z}_{\text{new},b}) \end{bmatrix}$$

being the $c \times b$ matrix of inner products between samples in \mathbf{Z}_R and those in \mathbf{Z}_{new} . Note that the matrix \mathbf{F}_R represents the virtual feature matrix of the original important samples along with the new important ones, but in the old embedding. Similarly, we could denote \mathbf{F}_{new} the virtual sample matrix of \mathbf{Z}_{new} as we do later in explaining the experiment in Section V.B.3

Now, let us denote the full set of important samples in the new embedding as $\tilde{\mathbf{F}}_R = \tilde{\Sigma}^{-1/2} \tilde{\mathbf{V}}^\top \tilde{\mathbf{W}} \in \mathbb{R}^{(c+b) \times (c+b)}$ where $\tilde{\mathbf{W}}$ is the kernel Gram matrix associated with augmented important sample set $\tilde{\mathbf{Z}}_R$. As mentioned before, the matrices $\tilde{\mathbf{V}}$ and $\tilde{\Sigma}$ need to be found for this new $\tilde{\mathbf{W}}$ matrix. Assuming that both the old and new important virtual sample matrices, \mathbf{F}_R and $\tilde{\mathbf{F}}_R$, have been found according to the method in Section II, the goal here is to find a transformation matrix $\mathbf{T} \in \mathbb{R}^{(c+b) \times c}$ that maps \mathbf{F}_R to $\tilde{\mathbf{F}}_R$ with minimum squared error.

To find the matrix \mathbf{T} , we assume that \mathbf{W} is full rank and $k = c$, i.e., we use all c eigenvectors in the original embedding of (2). The optimum mapping matrix \mathbf{T} is then obtained using

$$\mathbf{T}^* = \underset{\mathbf{T}}{\text{argmin}} \|\mathbf{T}\mathbf{F}_R - \tilde{\mathbf{F}}_R\|_F^2, \quad (15)$$

the solution of which gives

$$\begin{aligned} \mathbf{T}^* &= \tilde{\mathbf{F}}_R \mathbf{F}_R^\top (\mathbf{F}_R \mathbf{F}_R^\top)^{-1} \\ &= \Sigma^{-1/2} \tilde{\mathbf{V}}^\top \tilde{\mathbf{W}} \mathbf{W}_+^\top (\mathbf{W}_+ \mathbf{W}_+^\top)^{-1} \mathbf{V} \Sigma^{1/2}. \end{aligned} \quad (16)$$

Proposition 4.1: The minimum squared error $\|\mathbf{E}\|_F^2 = \|\mathbf{T}^* \mathbf{F}_R - \tilde{\mathbf{F}}_R\|_F^2$ measures the sum of eigenvalues of the updated kernel matrix $\tilde{\mathbf{W}}$ when projected onto $\langle \mathbf{W}_+^\top \rangle^\perp$, i.e., $\|\mathbf{E}\|_F^2 = \text{tr}(\mathbf{P}_{\mathbf{W}_+^\top}^\perp \tilde{\mathbf{W}})$.

Proof: First, let us expand the squared norm term,

$$\|\mathbf{E}\|_F^2 = \text{tr}(\mathbf{T}\mathbf{F}_R \mathbf{F}_R^\top \mathbf{T}^\top) - 2\text{tr}(\mathbf{T}\mathbf{F}_R \tilde{\mathbf{F}}_R^\top) + \text{tr}(\tilde{\mathbf{F}}_R \tilde{\mathbf{F}}_R^\top), \quad (17)$$

We then use the expression for the optimum \mathbf{T} in (16) and those of \mathbf{F}_R and $\tilde{\mathbf{F}}_R$ given previously. Also, we note that $\mathbf{F}_R^\top (\mathbf{F}_R \mathbf{F}_R^\top)^{-1} \mathbf{F}_R = \mathbf{P}_{\mathbf{F}_R^\top} = \mathbf{P}_{\mathbf{W}_+^\top}$ is a $c + b$ -dimensional projection matrix that projects onto the subspace spanned by \mathbf{F}_R^\top (or \mathbf{W}_+^\top).

Now, using the cyclic property of the trace, each term in the right hand side of (17) can be expressed as follows:

$$\begin{aligned} \text{tr}(\mathbf{T}\mathbf{F}_R \mathbf{F}_R^\top \mathbf{T}^\top) &= \text{tr}(\mathbf{F}_R^\top \mathbf{T}^\top \mathbf{T} \mathbf{F}_R) \\ &= \text{tr}(\mathbf{P}_{\mathbf{W}_+^\top} \tilde{\mathbf{F}}_R^\top \tilde{\mathbf{F}}_R \mathbf{P}_{\mathbf{W}_+^\top}) \\ &= \text{tr}(\mathbf{P}_{\mathbf{W}_+^\top} \tilde{\mathbf{W}}), \end{aligned}$$

$$\begin{aligned} \text{tr}(\mathbf{T}\mathbf{F}_R \tilde{\mathbf{F}}_R^\top) &= \text{tr}(\tilde{\mathbf{F}}_R \mathbf{P}_{\mathbf{W}_+^\top} \tilde{\mathbf{F}}_R^\top) \\ &= \text{tr}(\mathbf{P}_{\mathbf{W}_+^\top} \tilde{\mathbf{F}}_R^\top \tilde{\mathbf{F}}_R) \\ &= \text{tr}(\mathbf{P}_{\mathbf{W}_+^\top} \tilde{\mathbf{W}}), \end{aligned}$$

and

$$\begin{aligned} \text{tr}(\tilde{\mathbf{F}}_R \tilde{\mathbf{F}}_R^\top) &= \text{tr}(\tilde{\mathbf{F}}_R^\top \tilde{\mathbf{F}}_R) \\ &= \text{tr}(\tilde{\mathbf{W}}). \end{aligned}$$

Using the above results the minimum squared error can be expressed as

$$\|\mathbf{E}\|_F^2 = \text{tr}(\mathbf{P}_{\mathbf{W}_+^\top}^\perp \tilde{\mathbf{W}})$$

□

Algorithm 3 below provides the step-by-step procedure for the incremental dictionary updating process outlined in this section. The process involves incremental updating of the important sample set \mathbf{Z}_R , the virtual features \mathbf{F} , and the dictionary matrix \mathbf{H} . This assumes that there is a base system trained, and then new samples are presented to the system to improve the performance of the system without degrading performance on previously seen samples. This provides a way of updating the Nyström approximation of the full kernel matrix \mathbf{K} by selecting new important samples \mathbf{Z}_{new} to add to the set \mathbf{Z}_R , updating the eigenpairs (Σ, \mathbf{V}) given the new set $\tilde{\mathbf{Z}}_R$, transforming the previous dictionary \mathbf{H} to incorporate the information from the new samples, and determine if the error introduced through \mathbf{T} is above some threshold in which case we tune the transformed dictionary via IDU using the updated representations of the new important samples.

Remark 4.1: As pointed out before, the purpose of the transformation matrix \mathbf{T} is to provide an alternative approach to complete retraining using K-SVD when an embedding has been updated. In total this transform costs approximately $11c^3$ flops when computed via (16) plus an additional $2ML(c^2 + cb)$ operations to apply \mathbf{T} to each of the M old dictionaries. In its most efficient form [34], the K-SVD process costs approximately

Algorithm 3: Joint Embedding and Dictionary Updating.

Init: Generate baseline embedding components Σ, \mathbf{V} via method in Section II. Generate baseline dictionary \mathbf{H} via K-SVD learning using \mathbf{F} with all N baseline virtual samples.

Require: New important samples \mathbf{Z}_{new} . Current eigensystem Σ and \mathbf{V} ; Dictionary \mathbf{H} ; Error tolerance ε ; IDU algorithm [5] $\text{IDU}(\mathbf{F}, \mathbf{H}, K_1, \tau, n)$ with no. new atoms K_1 , sparsity factor τ and number of iterations n ;

Ensure: Important sample set $\tilde{\mathbf{Z}}_R$; Transformed dictionary $\tilde{\mathbf{H}}$; Eigenvalues $\tilde{\Sigma}$ and eigenvectors $\tilde{\mathbf{V}}$ of $\tilde{\mathbf{W}}$;

- 1: $\mathbf{Z}_R = [\mathbf{Z}_R, \mathbf{Z}_{new}]$
- 2: Update eigensystem (Σ, \mathbf{V}) as presented in Section III.
- 3: $\mathbf{C}_{new}^\top \leftarrow \mathbf{k}(\mathbf{Z}_R, \mathbf{Z}_{new})$
- 4: $\tilde{\mathbf{C}}_{new} \leftarrow [\mathbf{C}_{new}, \mathbf{k}(\mathbf{Z}_{new}, \mathbf{Z}_{new})]$
- 5: $\tilde{\mathbf{F}}_{new} \leftarrow \tilde{\Sigma}^{-1/2} \tilde{\mathbf{V}}^\top \tilde{\mathbf{C}}_{new}^\top$
- 6: $\mathbf{W}_+ \leftarrow [\mathbf{W}, \mathbf{C}_{new}^\top]$
- 7: $\tilde{\mathbf{W}} \leftarrow \begin{bmatrix} \mathbf{W}_+ \\ \tilde{\mathbf{C}}_{new} \end{bmatrix}$
- 8: $\mathbf{T} \leftarrow \tilde{\Sigma}^{-1/2} \tilde{\mathbf{V}}^\top \tilde{\mathbf{W}} \mathbf{W}_+^\top (\mathbf{W}_+ \mathbf{W}_+^\top)^{-1} \mathbf{V} \Sigma^{1/2}$.
- 9: $\tilde{\mathbf{H}}_m \leftarrow \mathbf{T} \mathbf{H}_m \quad \forall m \in [1, M]$
- 10: **if** error $e = \text{tr}(\mathbf{P}_{\tilde{\mathbf{W}}_+}^\perp \tilde{\mathbf{W}}) > \varepsilon$ **then**
- 11: $\tilde{\mathbf{H}} \leftarrow \text{IDU}(\tilde{\mathbf{F}}_{new}, \tilde{\mathbf{H}}, K_1, \tau, n)$
- 12: **end if**

$NL(2c\tau + (c\tau)^2)$ for a *single training iteration* where N, L, τ , and c are the number of training samples, the number of dictionary atoms, the sparsity factor, and the dimension of the input samples, respectively. To re-train each M dictionary matrix, this cost must be incurred I times, where I is the number of training iterations. Typically, when using Nyström $c \ll L < N$. As a result, it is more efficient to transform the dictionaries to the new embedding using the \mathbf{T} matrix rather than retraining using a repeated K-SVD process.

Remark 4.2: As stated before, dictionary updating/augmentation using the IDU method [5] results in expansion of the column space of dictionary matrices when informative samples are encountered in a new environment. Although this gives the system the ability to incorporate these new samples, the computational demand to do so grows with increasing dictionary sizes. Moreover, the likelihood of having more redundant basis vectors or dictionary atoms can lead to poor performance. To avoid having oversized dictionary matrices one can impose constraints to determine which dictionary atoms are essential in constructing the classification score and which atoms can be discarded without any detriment to the decision-making process.

A dictionary atom is *redundant* if it can be represented by other atoms. A dictionary atom is *inactive* if it doesn't participate actively in representing new and old samples. A likelihood weighted sampling that approximates the expected value of error over the training data by weighting each atom according to their importance can be devised to prune the redundant or inactive

atoms. Such methods have successfully been used before in importance weighting [35] for active learning.

V. EXPERIMENTAL RESULTS

The¹ experiments conducted in this paper use two handwritten digit image data sets, namely the MNIST and USPS. The MNIST database features 70,000 28×28 pixel grayscale images of handwritten single digits between 0 and 9. It uses 4 pixel zero-padding on all data samples and hence the actual characters reside in a 20×20 sub-image centered in the 28×28 image. The USPS is a smaller data set containing only 9298 16×16 grayscale images of handwritten digits 0-9. The images in this data set have no zero-padding and hence to rectify the size discrepancies, the images were first upsampled to size 20×20 pixels and then 4 pixel zero-padding was added to make them 28×28 pixel images.

Among the MNIST samples, 60,000 of these samples (or $\approx 86\%$ of the total) were treated as training samples and 10,000 of them were treated as testing samples, we refer to these groups as the MNIST training and testing sets, respectively. For the USPS samples, we set aside a small portion of the USPS data ($\leq 4\%$ of the USPS samples) for the experiments in the next section, this set of samples is henceforth referred to as the *in-situ* USPS samples. The testing set for USPS data set includes all of the USPS data *except* for the *in-situ* USPS samples.

Here, we consider three different experiments. In the first experiment, we demonstrate the performance of a baseline model which uses the embedding of Section II. The baseline dictionaries are trained via K-SVD using *only* virtual feature vectors extracted from the MNIST data set while testing is done on both the MNIST and USPS testing data sets. The other two experiments involve the SAU embedding update procedure in Section III using the baseline embedding and dictionaries generated in Experiment 1 as inputs. These experiments, however, differ in the approach they use to update the dictionary matrices. The second experiment will demonstrate the performance after an embedding update and full dictionary retraining using the K-SVD dictionary learning method. The results of this experiment will then be considered as the *benchmark* for the next experiment. In the third experiment an alternative to complete dictionary retraining after an embedding update is explored using the methods proposed in this paper. More specifically, dictionary matrices for the new embedding are first transformed via \mathbf{T} and *then* the incremental dictionary training (IDU) [5] is used to augment each of the transformed dictionaries with a limited number of atoms trained incrementally on a small portion of the USPS samples. This model is then tested on the same test set.

For each experiment, the following model parameters were selected. The baseline number of important samples for the Linearized Kernel Embedding (LKE) was chosen to be $c_0 = 1200$, and samples were chosen via uniform sampling scheme. When updating, $c_1 = 345$ additional "important" samples were

¹The authors have created a repository containing all scripts and modules required to reproduce the results of this work. This repository can be found at https://github.com/hilikliming/ILKE_SAU_paper, two Julia modules are included: EasySRC.jl and EasyLKE.jl

TABLE II
MNIST AND USPS CORRECT CLASSIFICATION RATES FOR 3 METHODS

Data set	Method	Cl. 1	Cl. 2	Cl. 3	Cl. 4	Cl. 5	Cl. 6	Cl. 7	Cl. 8	Cl. 9	Cl. 10	Avg. CC%
MNIST	Baseline	0.995	0.971	0.965	0.978	0.951	0.980	0.978	0.962	0.961	0.993	0.974
	K-SVD Retrain	0.996	0.986	0.959	0.978	0.959	0.977	0.981	0.956	0.964	0.991	0.975
	T+IDU	0.995	0.972	0.966	0.977	0.951	0.978	0.979	0.961	0.961	0.991	0.974
USPS	Baseline	0.992	0.954	0.967	0.871	0.789	0.879	0.276	0.869	0.195	0.983	0.807
	K-SVD Retrain	0.992	0.954	0.966	0.869	0.822	0.938	0.936	0.916	0.752	0.989	0.925
	T+IDU	0.994	0.960	0.969	0.881	0.825	0.947	0.949	0.907	0.804	0.983	0.932

selected from the USPS *in-situ* set. The rank of the baseline Nyström approximation, k , was chosen to be $k = 400$. A Gaussian kernel function $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2))$ with $\sigma = 2$ was chosen. A baseline dictionary was trained (Experiment 1) for each of the 10 digit classes with $L = 800$ and sparsity factor of $\tau = 10$. Dictionaries were trained to achieve a desired average squared error $\epsilon = 1e-7$ or a maximum of 30 training iterations per class dictionary. When using the dictionary transformation method, the full (i.e. c_0 -dimensional, then $c_0 + c_1$ -dimensional) matrices \mathbf{V} , $\tilde{\Sigma}$ and $\tilde{\mathbf{V}}$, $\tilde{\Sigma}$ were stored before and after model updates and were used when forming \mathbf{F} and $\tilde{\mathbf{F}}$ feature matrices. However, only the first k eigenpairs were used for the baseline model, and $k = 460$ was used for all embedding-updated models i.e. all models, except baseline, use 460-dimensional virtual features and $\mathbf{T} \in \mathbb{R}^{460 \times 400}$.

A. Different Experiments & Results

1) *Experiment 1-Baseline Training*: In this experiment, a baseline model is trained via the K-SVD dictionary learning method using all of the virtual feature vectors generated for the MNIST training data set and then tested on those of the MNIST and USPS test sets. Table II illustrates the correct classification rates (CCR) for each of the 10 classes in the MNIST and USPS testing sets, respectively. As expected, the baseline model performed well on MNIST with an overall CCR of 97.4%. The performance of the baseline model on the USPS test set, however, was comparatively lower (see row 4 of Table II). Nevertheless, it still managed to achieve an overall CCR of 80.7%, despite there being no training samples from this data set.

The baseline embedding and dictionary matrices, i.e., Σ , \mathbf{V} and $\mathbf{H}_1, \dots, \mathbf{H}_{10}$, generated in this experiment were used as inputs to all of the subsequent experiments. However, as mentioned before the next experiments utilize the SAU embedding update method presented in Section III and the same set of *in-situ* USPS samples, though not all samples were necessarily selected as important samples.

2) *Experiment 2-Benchmark Method*: The experiment in this section utilizes resources for complete model/dictionary retraining. To this end, the baseline embedding and dictionaries generated in Experiment 1 are used for the embedding update via our SAU algorithm in Section III using the *in-situ* USPS virtual samples. The K-SVD dictionary learning is then applied to fully retrain each of the dictionary matrices using *all* the training samples including the newly embedded *in-situ* samples (i.e. baseline and *in-situ* samples combined). The embedding updated and retrained model was then tested on the MNIST and USPS test sets and results were compared to those of the other experiments. Furthermore, we show the ability of the system to

retain performance on the original baseline environment after updating the embedding and dictionaries.

The CCR for each class using the K-SVD retrained model are shown in rows 2 and 5 of Table II for the MNIST and USPS test sets, respectively. On the MNIST testing set, this K-SVD retrain model achieved an overall CCR of 97.5%. As far as the results on the USPS test set are concerned this K-SVD retrained model provided significant performance improvement. More specifically, using $\leq 4\%$ of the total USPS samples, the modified MSC with the updated dictionaries were able to achieve an overall CCR of 92.5%, i.e., more than a 10% improvement over that of the baseline trained system.

3) *Experiment 3-Joint Embedding & Dictionary Updating*: The experiment in this section attempts to demonstrate a contrast between dictionary transformation using \mathbf{T} matrix and dictionary updating, and full K-SVD retraining of the dictionaries in the updated embedding. As before, the *in-situ* USPS samples were used to update the baseline embedding and the \mathbf{T} matrix was generated. Using \mathbf{T} , we transform each of the old dictionaries to the new embedding. Then, following Algorithm 3, we augment the \mathbf{T} -transformed dictionaries with $K_1 = 10$ new atoms trained using the IDU method in [5] with $\tilde{\mathbf{F}}_{\text{new}} \subset \tilde{\mathbf{F}}$ representations of *only* the *in-situ* USPS samples, including the c_1 samples that were added to $\tilde{\mathbf{F}}_R \subset \tilde{\mathbf{F}}_{\text{new}}$.

All other training parameters aside from K_1 , e.g., sparsity factor, error goal, etc., were kept the same as in the baseline training. The results of this model, which combines transforming old dictionaries and training a limited number of new atoms, can be seen in rows 3 and 6 of Table II for the MNIST and USPS test sets, respectively. While the performance observed on the MNIST test data is still comparable to that of the baseline results in Experiment 1, classification accuracy improves across almost every class in the USPS test set. Although, the results on the MNIST test set are slightly worse than those of the fully retrained system, the results on the USPS test set are indeed comparable to those generated using the fully retrained K-SVD model in Experiment 2. The overall CCRs on the MNIST and USPS test sets are found to be 97.4% and 93.2%, respectively.

VI. CONCLUSION

In this paper, we proposed a novel method for incrementally updating the linearized kernel embedding in the LKDL method. We presented a new mechanism to incrementally perform eigen-decomposition of a growing kernel matrix using arrowhead updates. We also made efforts to shed light on the properties of the embedding itself and its geometric interpretation. A new dictionary updating method was presented for LKDL dictionaries which allowed them to be transformed to the expanded embedding without the need for new labels. This method presents an alternative to the complete repetitive dictionary retraining

which can be computationally costly and slow. A modified MSC classifier was used in conjunction with a variety of class-dependent dictionaries to perform classification on the MNIST and USPS handwritten digit data sets in conjunction with an incremental learning strategy. Results were provided comparing the classification performance of a baseline dictionary, dictionaries retrained after expansion of the embedding, and lastly using dictionaries updated through our proposed method after expansion of the embedding. These results indicated that the newly developed incremental linearized kernel embedding can effectively incorporate samples from novel environments to allow SRC-type classifiers to adapt to the new environments.

The computational time and consistency of the methods for eigendecomposition of a growing kernel matrix were also studied and compared. The results showed: (a) when a model is expected to undergo many embedding updates during its lifetime, the arrowhead approach presented in this paper solves the new embedding in the most competitive time; (b) dictionary matrices in the new embedding are comparable to those of the old embedding, and can be obtained by using a simple projection of the old dictionary; and (c) combination of transforming dictionary matrices and training a limited number of incremental atoms to augment the dictionary provides comparable, if not superior, performance to the fully retraining system via K-SVD with the same training parameters.

Looking forward, there are extensions to our work that can provide further computational and discriminative improvements. Among these is the development of a down-dating method which leverages arrowhead eigendecompositions to complement our proposed SAU. A down-dating method of this variety would enable efficient pruning of growing embeddings, allowing for a static embedding dimension while maintaining the fast updates of the proposed method. The down-dating method would require the *removal* of important samples from the formation of the embedding, thus a mechanism for performing the important sample selection in both the up- and down-dating methods would be preferable. Future efforts could be focused on this important sample selection process using various metrics.

REFERENCES

- [1] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
- [2] A. Salberg, A. Hanssen, and L. Scharf, "Robust multidimensional matched subspace classifiers based on weighted least-squares," *IEEE Trans. Signal Process.*, vol. 55, no. 3, pp. 873–880, Mar. 2007.
- [3] A. Golts and M. Elad, "Linearized kernel dictionary learning," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 4, pp. 726–739, Jun. 2016.
- [4] L. Wang, K. Lu, P. Liu, R. Ranjan, and L. Chen, "IK-SVD: Dictionary learning for spatial Big Data via incremental atom update," *Comput. Sci. Eng.*, vol. 16, no. 4, pp. 41–52, 2014.
- [5] M. R. Azimi-Sadjadi, Y. Zhao, and S. Sheedvash, "Incremental dictionary learning with sparsity," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2018, pp. 1–6.
- [6] M. R. Azimi-Sadjadi, C. Robbiano, Y. Zhao, and J. J. Hall, "Incremental dictionary learning for adaptive classification and reconstruction of facial imagery," in *Proc. IEEE 29th Int. Workshop Mach. Learn. Signal Process.*, 2019, pp. 1–6.
- [7] Y. C. Pati, R. Rezaifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. IEEE 27th Asilomar Conf. Signals, Systems Comput.*, 1993, pp. 40–44.
- [8] M. R. Azimi-Sadjadi, J. Kopicz, and N. Klausner, "K-SVD dictionary learning using a fast OMP with applications," in *Proc. IEEE Int. Conf. Image Process.*, 2014, pp. 1599–1603.
- [9] S. Kargl, A. España, K. Williams, J. Kennedy, and J. Lopes, "Scattering from objects at a water-sediment interface: Experiment, high-speed and high-fidelity models, and physical insight," *IEEE J. Ocean. Eng.*, vol. 40, no. 3, pp. 632–642, Jul. 2015.
- [10] J. Bucaro *et al.*, "Acoustic identification of buried underwater exploded ordnance using a numerically trained classifier," *J. Acoustical Soc. Amer.*, vol. 132, pp. 3614–3617, Dec. 2012.
- [11] J. J. Hall, M. R. Azimi-Sadjadi, S. G. Kargl, Y. Zhao, and K. L. Williams, "Underwater unexploded ordnance (UXO) classification using a matched subspace classifier with adaptive dictionaries," *IEEE J. Ocean. Eng.*, no. 99, no. 3, pp. 1–14, Jul. 2019.
- [12] S. Dey *et al.*, "Structural-acoustic modeling for three-dimensional freefield and littoral environments with verification and validation," *J. Acoustical Soc. Amer.*, vol. 129, pp. 2979–2990, May 2011.
- [13] D. P. Williams, "On the use of tiny convolutional neural networks for human-expert-level classification performance in sonar imagery," *IEEE J. Ocean. Eng.*, vol. 46, no. 1, pp. 236–260, Jan. 2021.
- [14] F. Hallgren and P. Northrop, "Incremental kernel PCA and the nyström method," 2018, *arXiv:1802.00043*.
- [15] N. J. Stor, I. Slapnicar, and J. L. Barlow, "Accurate eigenvalue decomposition of real symmetric arrowhead matrices and applications," *Linear Algebra its Appl.*, vol. 464, pp. 62–89, 2015.
- [16] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, 1998.
- [17] C. Musco and C. Musco, "Recursive sampling for the nyström method," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3833–3845.
- [18] C. K. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Adv. Neural Inf. Process. Syst.*, 2001, pp. 682–688.
- [19] B. Schölkopf and A. Smola, *Learning With Kernels*. Cambridge, MA, USA: MIT press, pp. 427–437, 2002.
- [20] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the nyström method," *J. Mach. Learn. Res.*, vol. 13, pp. 981–1006, 2012.
- [21] J. Shawe-Taylor, C. Williams, N. Cristianini, and J. Kandola, "On the eigenspectrum of the gram matrix and the generalization error of kernel-PCA," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2510–2522, Jul. 2005.
- [22] R. D. DeGroat and R. A. Roberts, "Efficient, numerically stabilized rank-one eigenstructure updating (signal processing)," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 2, pp. 301–316, Feb. 1990.
- [23] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, no. 1, pp. 31–48, 1978.
- [24] G. H. Golub, "Some modified matrix eigenvalue problems," *Siam Rev.*, vol. 15, no. 2, pp. 318–334, 1973.
- [25] G. H. Golub and C. F. Van Loan, *Matrix Computations*, vol. 3. Baltimore, MD, USA: JHU Press, 2013.
- [26] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Mach. Learn.*, vol. 73, no. 3, pp. 243–272, 2008.
- [27] P. Ruvolo and E. Eaton, "ELLA: An efficient lifelong learning algorithm," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, vol. 28, pp. 507–515.
- [28] M.-F. Balcan, A. Blum, and S. Vempala, "Efficient representations for lifelong learning and autoencoding," in *Proc. Conf. Learn. Theory*, 2015, pp. 191–210.
- [29] D. Isele, M. Rostami, and E. Eaton, "Using task features for zero-shot knowledge transfer in lifelong learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 1620–1626.
- [30] M. Rostami, D. Isele, and E. Eaton, "Using task descriptions in lifelong machine learning for improved performance and zero-shot transfer," *J. Artif. Intell. Res.*, vol. 67, pp. 673–704, 2020.
- [31] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.
- [32] S. Mika *et al.*, "Kernel PCA and de-noising in feature spaces," in *Adv. Neural Inf. Process. Syst.*, vol. 11, 1998, pp. 536–542.
- [33] J.-Y. Kwok and I.-H. Tsang, "The pre-image problem in kernel methods," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1517–1525, Nov. 2004.
- [34] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," *Comput. Sci. Dept., Technion, Tech. Rep. CS-2008-08*, 2008.
- [35] A. Beygelzimer, S. Dasgupta, and J. Langford, "Importance weighted active learning," in *Proc. ICML*, 2009, pp. 49–56.



and machine learning.

John Joseph Hall (Member, IEEE) received the B.S. degree in electrical engineering from Miami University, Oxford, OH, USA, in 2014 and the M.S. degree in electrical and computer engineering from Colorado State University, Fort Collins, CO, USA, in 2016. He defended his dissertation at Colorado State University where his research focused on lifelong learning for sonar mine-hunting problems. He is currently a Senior AI Engineer with Trimble PPM. His research interests include digital and statistical signal and image processing, adaptive filtering, artificial intelligence,



processing, machine learning and adaptive systems, target detection, classification and tracking, sensor array processing, and distributed sensor networks.

Dr. Azimi-Sadjadi was an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the IEEE TRANSACTIONS ON NEURAL NETWORKS.

Mahmood R. Azimi-Sadjadi (Life Member, IEEE) received the M.S. and Ph.D. degrees from the Imperial College of Science and Technology, University of London, London, U.K., in 1978 and 1982, respectively, in electrical engineering with specialization in digital signal/image processing. He is currently a Full Professor with the Electrical and Computer Engineering Department, Colorado State University (CSU), Fort Collins, CO, USA. He is also the Director of the Digital Signal/Image Laboratory, CSU. His research interests include statistical signal and image



research interests include machine learning, autonomous systems, and statistical signal processing.

Christopher Robbiano (Member, IEEE) received the B.S. degree in physics and M.S. degree in electrical engineering from Colorado State University, Fort Collins, CO, USA, in 2011 and 2017, respectively. He defended his dissertation with Colorado State University where his research has been in the areas of interactive sensing for sonar systems. He is currently a Senior Systems Engineer with Ball Aerospace & Technologies Corporation in Westminster Company, and previously was a Research Scientist with Information System Technologies Incorporated. His