Optimizing Kernel Machines Using Deep Learning

Huan Song, *Member, IEEE,* Jayaraman J. Thiagarajan, *Member, IEEE,* Prasanna Sattigeri, *Member, IEEE,* and Andreas Spanias, *Fellow, IEEE*

***Abstract*— Building highly nonlinear and nonparametric models is central to several state-of-the-art machine learning systems. Kernel methods form an important class of techniques that induce a reproducing kernel Hilbert space (RKHS) for inferring non-linear models through the construction of similarity functions from data. These methods are particularly preferred in cases where the training data sizes are limited and when prior knowledge of the data similarities is available. Despite their usefulness, they are limited by the computational complexity and their inability to support end-to-end learning with a task-specific objective. On the other hand, deep neural networks have become the de facto solution for end-to-end inference in several learning paradigms. In this paper, we explore the idea of using deep architectures to perform kernel machine optimization, for both computational efficiency and end-to-end inferencing. To this end, we develop the deep kernel machine optimization framework, that creates an ensemble of dense embeddings using Nyström kernel approximations and utilizes deep learning to generate task-specific representations through the fusion of the embeddings. Intuitively, the filters of the network are trained to fuse information from an ensemble of linear subspaces in the RKHS. Furthermore, we introduce the kernel dropout regularization to enable improved training convergence. Finally, we extend this framework to the multiple kernel case, by coupling a global fusion layer with pretrained deep kernel machines for each of the constituent kernels. Using case studies with limited training data, and lack of explicit feature sources, we demonstrate the effectiveness of our framework over conventional model inferencing techniques.**

***Index Terms*— Deep neural networks (DNNs), kernel methods, multiple kernel learning (MKL), Nyström approximation.**

I. Introduction

THE recent surge in representation learning for complex, high-dimensional data has revolutionized machine learning and data analysis. The success of deep neural networks (DNNs) in a wide variety of computer vision tasks has emphasized the need for highly nonlinear and nonparametric models [1], [2]. In particular, by coupling modern deep architectures with large data sets [3], [4], efficient optimization strategies [5], [6], and GPU utilization, one can obtain highly effective predictive models. By using a composition of multiple non-linear transformations, along with novel loss functions, DNNs can approximate a large class of functions for prediction tasks. However, the increasing complexity of the networks requires exhaustive tuning of several hyper-parameters in the discrete space of network architectures, often resulting in suboptimal solutions or model overfitting. This is particularly more common in applications characterized by limited data set sizes and complex dependencies in the input space. Despite the advances in regularization techniques and data augmentation strategies [7], in many scenarios, it is challenging to obtain deep architectures that provide significant performance improvements over conventional machine learning solutions. In such cases, a popular alternative solution to building effective, nonlinear predictive models is to employ kernel machines.

Manuscript received June 24, 2017; revised November 14, 2017; accepted January 30, 2018. Date of publication March 6, 2018; date of current version October 16, 2018. This work was supported in part by the SenSIP Center at Arizona State University and in part by the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. *(Corresponding author: Huan Song.)*

H. Song and A. Spanias are with the SenSIP Center, School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: huan.song@asu.edu; spanias@asu.edu).

J. J. Thiagarajan is with the Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (e-mail: jjayaram@llnl.gov).

P. Sattigeri is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: psattig@us.ibm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2804895

A. Kernel Methods and Multiple Kernel Learning

Kernel methods have a long-standing success in machine learning, primarily due to their well-developed theory, convex formulations, and their flexibility in incorporating prior knowledge of the dependencies in the input space. Denoting the $d-$dimensional input domain as $\mathcal{X} \subset \mathbb{R}^d$, the kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ induces an implicit mapping into a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$, through the construction of a positive definite similarity matrix between samples in the input space. An appealing feature of this approach is that even simple linear models inferred in the RKHS are highly effective compared to their linear counterparts learned directly in the input space.

Kernel methods are versatile in that specifying a positive-definite kernel will enable the use of this generic optimization framework for any data representation, such as vectors, matrices, sequences or graphs. Consequently, a broad range of kernel construction strategies have been proposed in the literature, e.g., χ^2 kernel [8], string [9], and graph kernels [10]. Furthermore, the classical Representer Theorem allows the representation of any optimal function in $\mathcal{H}_k$ thereby enabling construction of a dual optimization problem based only on the kernel matrix and not the samples explicitly. This is commonly referred as the *kernel trick* in the machine learning literature. Finally, kernel methods can be augmented with a variety of strategies for controlling the learning capacity and hence reducing model overfitting [11].

Despite these advantages, kernel methods have some crucial limitations when applied in practice: (a) The first limitation is their computational complexity, which grows quadratically with the sample size due to the computation of the kernel (Gram) matrix. A popular solution to address this challenge is to approximate the kernel matrix using the Nyström method [12] or the random Fourier features based methods for shift-invariant kernels [13]. While the Nyström method obtains a low-rank approximation of the kernel matrix, the latter explicitly maps the data into an Euclidean inner product space using randomized feature maps; (b) Another crucial limitation of kernel methods is that, unlike the state-of-the-art deep learning systems, the data representation and model learning stages are decoupled and hence cannot admit end-to-end learning.

The active study in multiple kernel learning (MKL) alleviates this limitation to some extent. MKL algorithms [14] attempt to automatically select and combine multiple base kernels to exploit the complementary nature of the individual feature spaces and thus improve performance. A variety of strategies can be used for combining the kernel matrices, such that the resulting matrix is also positive definite, i.e., a valid kernel. Common examples include nonnegative sum [15] or hadamard product of the matrices [16]. Although MKL provides additional parameters to obtain an optimal RKHS for effective inference, the optimization (dual) is computationally more challenging, particularly with the increase in the number of kernels. More importantly, in practice, this optimization does not produce consistent performance improvements over a simple baseline kernel constructed as the unweighted average of the base kernels [17], [18]. Furthermore, extending MKL techniques, designed primarily for binary classification, to multiclass classification problems is not straightforward. In contrast to the conventional one-versus-rest approach, which decomposes the problem into multiple binary classification problems, in MKL it is beneficial to obtain the weighting of base kernels with respect to all classes [19], [20].

B. Bridging Deep Learning and Kernel Methods

In this paper, our goal is to utilize deep architectures to facilitate improved optimization of kernel machines, particularly in scenarios with limited labeled data and prior knowledge of the relationships in data (e.g., biological data sets). Existing efforts on bridging deep learning with kernel methods focus on using kernel compositions to emulate neural network layer stacking or enabling the optimization of deep architectures with data-specific kernels [21]. Combining the advantages of these two paradigms of predictive learning has led to new architectures and inference strategies. For example, in [22] and [23], the authors utilized kernel learning to define a new type of convolutional networks and demonstrated improved performance in inverse imaging problems (details in Section II-B).

Inspired by these efforts, in this paper, we develop a deep learning based solution to kernel machine optimization, for both single and multiple kernel cases. While existing kernel approximation techniques make kernel learning efficient, utilizing deep networks enables end-to-end inference with a task-specific objective. In contrast to approaches such as [22] and [23], which replace the conventional neural network operations, e.g., convolutions, using equivalent computations in the RKHS, we use the similarity kernel to construct dense embeddings for data and build task-specific representations, through fusion of these embeddings. Consequently, our approach is applicable to any kind of data representation. Similar to conventional kernel methods, our approach exploits the native space of the chosen kernel during inference, thereby controlling the capacity of learned models, and thus leading to improved generalization. Finally, in scenarios where multiple kernels are available during training, either corresponding to multiple feature sources or from different kernel parameterizations, we develop a multiple kernel variant of the proposed approach. Interestingly, in scenarios with limited amounts of data and in applications with no access to explicit feature sources, the proposed approach is superior to state-of-the-practice kernel machine optimization techniques and deep feature fusion techniques.

The main contributions of this paper can be summarized as follows:

- We develop deep kernel machine optimization (DKMO), which creates dense embeddings for the data through projection onto a subspace in the RKHS and learns task-specific representations using deep learning.
- To improve the effectiveness of the representations, we propose to create an ensemble of embeddings obtained from Nyström approximation methods, and pose the representation learning task as deep feature fusion.
- We introduce the kernel dropout regularization to enable robust feature learning with kernels from limited data.
- We develop M-DKMO, a multiple kernel variant of the proposed algorithm, to effectively perform MKL with multiple feature sources or kernel parameterizations.
- We show that on standardized data sets, where kernel methods have had proven success, the proposed approach outperforms the state-of-the-practice kernel methods, with a significantly simpler optimization.
- Using cell biology data sets, we demonstrate the effectiveness of our approach in cases where we do not have access to features but only encoded relationships.
- Under the constraint of limited training data, we show that our approach outperforms both the state-of-the-art MKL methods and standard DNNs applied to the feature sources directly.

II. Related Works

In this section, we briefly review the prior art in optimizing kernel machines and discuss the recent efforts toward bridging kernel methods and deep learning.

A. Kernel Machine Optimization

The success of kernel support vector machines (SVMs) [24] motivated the kernelization of a broad range of linear machine learning formulations in the Euclidean space. Popular examples are regression [25], clustering [26], unsupervised and supervised dimension reduction algorithms [27], dictionary learning for sparse representations [28], [29] and many others. Following the advent of more advanced data representations in machine learning algorithms, such as graphs and points on embedded manifolds, kernel methods provided a flexible framework to perform statistical learning with such data. Examples include the large class of graph kernels [10] and Grassmannian kernels for Riemannian manifolds of linear subspaces [30].

Despite the flexibility of this approach, the need to deal with kernel matrices makes the optimization infeasible in large scale data. There are two classes of approaches commonly used by researchers to alleviate this challenge. First, kernel approximation strategies can be used to reduce both computational and memory complexity of kernel methods, e.g., the Nyström method [12]. The crucial component in Nyström kernel approximation strategies is to select a subset of the kernel matrix to recover the inherent relationships in the data. A straightforward uniform sampling on columns of kernel matrix has been demonstrated to provide reasonable performance in many cases [31]. However, Zhang and Kwok [32] proposed an improved variant of Nyström approximation, that employs distance-based clustering to obtain landmark points in order to construct a subspace in the RKHS. Interestingly, the authors proved that the approximation error is bounded by the quantization error of coding each sample using its closest landmark. Kumar *et al.* [33] generated an ensemble of approximations by repeating Nyström random sampling multiple times for improving the quality of the approximation.

Second, in the case of shift-invariant kernels, random Fourier features can be used to design scalable kernel machines [34], [35]. Instead of using the implicit feature mapping in the kernel trick, Rahimi and Recht [34] proposed to utilize randomized features for approximating kernel evaluation. The idea is to explicitly map the data to an Euclidean inner product space using randomized feature maps, such that kernels can be approximated using Euclidean inner products. Using random Fourier features, Huang *et al.* [36] showed that shallow kernel machines matched the performance of deep networks in speech recognition, while being computationally efficient.

1) Combining Multiple Kernels: A straightforward extension to kernel learning is to consider multiple kernels. Here, the objective is to learn a combination of base kernels $k_1, \ldots, k_M$ and perform empirical risk minimization simultaneously. Conical [15] and convex combinations [37] are commonly considered and efficient optimizers such as sequential minimal optimization [15] and spectral projected gradient [38] techniques have been developed. In an extensive review of MKL algorithms, Gönen and Alpayd?n [14] showed that the formulation in [18] achieved consistently superior performance on several binary classification tasks. MKL algorithms have been applied to a wide range of machine learning problems. With base kernels constructed from distinct features, MKL can be utilized as a feature fusion mechanism [17], [39]–[42]. When base kernels originate from different feature sources or kernel parameterizations, MKL automates the kernel selection and parameter tuning process [15], [20]. Most recent research in MKL focus on improving the multi-class classification performance [20] and effectively handling training convergence and complexity [43].

These simple fusion schemes have been generalized further to create localized MKL (LMKL) [44]–[46] and nonlinear MKL algorithms. Moeller *et al.* [46] have formulated a unified view of LMKL algorithms

$$k_\beta(\mathbf{x}_i, \mathbf{x}_j) = \sum_m \beta_m(\mathbf{x}_i, \mathbf{x}_j) k_m(\mathbf{x}_i, \mathbf{x}_j) \tag{1}$$

where β_m is the gating function for kernel function k_m. In contrast to "global" MKL formulations where the weight β_m is constant across data, the gating function in (1) takes the data sample as an independent variable and is able to characterize the underlying local structure in data. Several LMKL algorithms differ in how β_m is constructed. For example, in [44], β_m is chosen to be separable into softmax functions. On the other hand, nonlinear MKL algorithms are based on the idea that non-linear combination of base kernels could provide richer and more expressive representations compared to linear mixing. For example, [18] considered polynomial combination of base kernels and [47] utilized a two-layer neural network to construct an radial basis function (RBF) kernel composition on top of the linear combination.

B. Combining Deep Learning With Kernel Methods

While the recent focus of research in kernel learning has been toward scaling kernel optimization and MKL, there is another important direction aimed at improving the representation power of kernel machines. In particular, inspired by the exceptional power of deep architectures in feature design and end-to-end learning, a recent wave of research efforts attempt to incorporate ideas from deep learning into kernel machine optimization [23], [47]–[50]. One of the earliest approaches in this direction was developed by Cho *et al.* [48], in which a new arc-cosine kernel was defined. Based on the observation that arc-cosine kernels possess characteristics similar to an infinite single-layer threshold network, the authors proposed to emulate the behavior of DNN by composition of arc-cosine kernels. The kernel composition idea using neural networks was then extended to MKL by Zhuang *et al.* [47]. The connection between kernel learning and deep learning can also be drawn through Gaussian processes as demonstrated by Wilson *et al.* [50], where theyderived deep kernels through the Gaussian process marginal likelihood. Another class of approaches directly incorporated kernel machines into DNN architectures. For example, Wiering *et al*. [49] constructed a multilayer SVM by replacing neurons in multilayer perceptrons) with SVM units. More recently, in [23], kernel approximation is carried out using supervised subspace learning in the RKHS, and backpropagation-based training similar to convolutional neural network (CNN) is adopted to optimize the parameters. The experimental results on image reconstruction and superresolution showed that the new type of network achieved competitive and sometimes improved performance as compared to CNN.

In this paper, we provide an alternative viewpoint to kernel machine optimization by considering the kernel approximate mappings as embeddings of the data and employ DNNs to infer task-specific representations as a fusion of an ensemble of subspace projections in the RKHS. Crucial advantages of

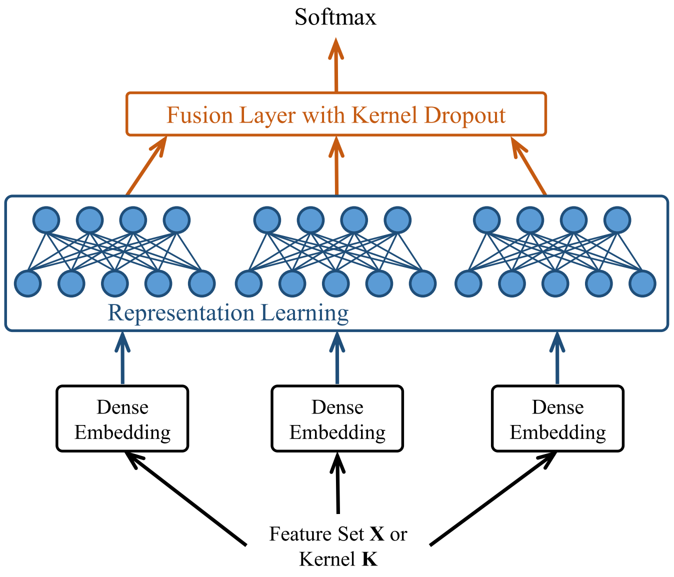


Fig. 1. DKMO—the proposed approach for optimizing kernel machines using DNNs. For a given kernel, we generate multiple dense embeddings using kernel approximation techniques, and fuse them in a fully connected DNN. The architecture utilizes fully connected networks with kernel dropout regularization during the fusion stage. Our approach can handle scenarios when both the feature sources and the kernel matrix are available during training or when only the kernel similarities can be accessed.

our approach are that extension to the multiple kernel case is straightforward, and it can be highly robust to smaller data sets.

III. Deep Kernel Machine Optimization—Single Kernel Case

In this section, we describe the proposed DKMO framework which utilizes the power of deep architectures in end-to-end learning and feature fusion to facilitate kernel learning. Viewed from bottom to top in Fig. 1, the DKMO first extracts multiple dense embeddings from a precomputed similarity kernel matrix $\mathbf{K}$ and optionally the feature source $\mathcal{X}$ if accessible during training. On top of each embedding, we build a fully connected neural network for representation learning. Given the inferred latent spaces from representation learning, we stack a fusion layer which is responsible for combining the latent features and obtaining a concise representation for inference tasks. Finally, we use a softmax layer at the top to perform classification, or an appropriate dense layer for regression tasks. Note that, similar to random Fourier feature based techniques in kernel methods, we learn a mapping to the Euclidean space, based on the kernel similarity matrix. However, in contrast, the representation learning phase is not decoupled from the actual task, and hence can lead to higher fidelity predictive models.

A. Dense Embedding Layer

From Fig. 1, it can be seen that the components of representation learning and fusion of hidden features are generic, i.e., they are separate from the input data or the kernel. Consequently, the dense embedding layer is the key component that bridges kernel representations with the DNN training, thereby enabling an end-to-end training.

1) Motivation: Consider the kernel Gram matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$, where $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. The j-th column encodes the relevance between sample $\mathbf{x}_j$ to all other samples $\mathbf{x}_i$ in the training set, and hence this can be viewed as an embedding for $\mathbf{x}_j$. As a result, these naive embeddings can potentially be used in the input layer of the network. However, $\mathbf{k}_j$ has large values at location corresponding to training samples belonging to the same class as $\mathbf{x}_j$ and small values close to zero at others. The sparsity and high dimensionality of these embeddings make them unsuitable for inference tasks.

A natural approach to alleviate this challenge is to adopt kernel matrix factorization strategies, which transform the original embedding into a more tractable, low-dimensional representation. This procedure can be viewed as kernel approximation with truncated SVD or Nyström methods [12]. Furthermore, this is conceptually similar to the process of obtaining dense word embeddings in natural language processing. For example, Levy and Goldberg [51] have shown that the popular skip-gram with negative sampling (SGNS) model in language modeling is implicitly factorizing the pointwise mutual information matrix, whose entries measure the association between pairs of words. Interestingly, they demonstrated that alternate word embeddings obtained using the truncated SVD method are more effective than SGNS on some word modeling tasks [51].

In existing deep kernel learning approaches such as the convolutional kernel networks [23], the key idea is to construct multiple RKHSs at different convolutional layers of the network, with a sequence of pooling operations between the layers to facilitate kernel design for different subregion sizes. However, this approach cannot generalize to scenarios where the kernels are not constructed from images, for example, in the case of biological sequences. Consequently, we propose to obtain multiple approximate mappings (dense embeddings) from the feature set or the kernel matrix using Nyström methods, and then utilize the DNN as both representation learning and feature fusion mechanisms to obtain a task-specific representation for data in the Euclidean space. All components in this framework are general and are not constrained by the application or kind of data used for training.

2) Dense Embeddings Using Nyström Approximation: In order to be flexible with different problem settings, we consider two different pipelines for constructing the dense embeddings based on Nyström approximation: I) In many applications, e.g., biological sequences or social networks, it is often easier to quantify sample-to-sample distance or similarity than deriving effective features or measurements for inference tasks. Furthermore, for many existing data sets, large-scale pairwise distances are already precomputed and can be easily converted into kernel matrices. In such scenarios, we use the conventional Nyström method to calculate the dense embeddings. II) When the input data is constructed from predefined feature sources, we employ the clustered Nyström method [32], which identifies a subspace in the RKHS using distance-based clustering, and explicitly project the feature mappings onto subspaces in the RKHS. In this case, the dense embeddings are obtained without constructing the complete kernel matrix for the data set. Next, we discuss these two strategies in detail.

a) Conventional Nyström approximation on kernels: In applications where the feature sources are not directly accessible, we construct dense embeddings from the kernel matrix directly. Based on the Nyström method [31], [33], a subset of s columns selected from $\mathbf{K}$ can be used to find an approximate kernel map $\mathbf{L} \in \mathbb{R}^{n\times r}$, such that $\mathbf{K} \simeq \mathbf{L}\mathbf{L}^T$ where $s \ll n$ and $r \leq s$. To better facilitate the subsequent DNN representation learning, we extract multiple approximate mappings through different random samplings of the kernel matrix. More specifically, from $\mathbf{K}$, we randomly select $s \times P$ columns without replacement, and then divide it into P sets containing s columns each. Consider a single set $\mathbf{E} \in \mathbb{R}^{n\times s}$ containing the selected columns and denote $\mathbf{W} \in \mathbb{R}^{s\times s}$ as the intersection of the selected columns and corresponding rows on $\mathbf{K}$. The rank-r approximation $\tilde{\mathbf{K}}_r$ of $\mathbf{K}$ is computed as

$$\tilde{\mathbf{K}}_r = \mathbf{E}\tilde{\mathbf{W}}_r\mathbf{E}^T \tag{2}$$

where $\tilde{\mathbf{W}}_r$ is the optimal rank-r approximation of $\mathbf{W}$ obtained using truncated SVD. As it can be observed, the time complexity of the approximation reduces to $O(s^3)$, which corresponds to performing SVD on $\mathbf{W}$. This can be further reduced by randomized SVD algorithms as shown in [52]. The approximate mapping function $\mathbf{L}$ can then be obtained by

$$\mathbf{L} = \mathbf{E}\big(\mathbf{U}_{\tilde{\mathbf{W}}_r}\Lambda_{\tilde{\mathbf{W}}_r}^{-1/2}\big) \tag{3}$$

where $\mathbf{U}_{\tilde{\mathbf{W}}_r}$ and $\Lambda_{\tilde{\mathbf{W}}_r}$ are top r eigenvalues and eigenvectors of $\mathbf{W}$.

With different sampling sets spanning distinct subspaces, the projections will result in completely different representations in the RKHS. Since the performance of our end-to-end learning approach is heavily influenced by the construction of subspaces in the RKHS, we propose to infer an ensemble of multiple subspace approximations for a given kernel. This is conceptually similar to [33], in which an ensemble of multiple Nyström approximations are inferred to construct an approximation of the kernel. However, our approach works directly with the approximate mappings $\mathbf{L}$ instead of approximated kernels $\tilde{\mathbf{K}}_r$ and the mappings are further coupled with the DNN optimization. The differences in the representations of the projected features will be exploited in the deep learning fusion architecture to model the characteristics in different regions of the input space. To this end, we repeat the calculation based on (3) for all P selected sets and obtain the dense embeddings $\mathbf{L}_1, \ldots, \mathbf{L}_P$.

b) Clustered nyström approximation on feature sets: When the feature sources are accessible, we propose to employ clustered Nyström approximation to obtain the dense embeddings directly from features without construction of the actual kernel. Following the approach in [32], k-means cluster centroids can be utilized as the set of the landmark points from $\mathbf{X}$. Denoting the matrix of landmark points by $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_r]$ and the subspace they span by $\mathcal{F} = \text{span}(\varphi(\mathbf{z}_1), \ldots, \varphi(\mathbf{z}_r))$, the projection of the samples $\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_n)$ in $\mathcal{H}_k$ onto its subspace $\mathcal{F}$ is equivalent to the following Nyström approximation (we refer to [23] for the detailed derivation):

$$\mathbf{L}_\mathbf{Z} = \mathbf{E}_\mathbf{Z}\mathbf{W}_\mathbf{Z}^{-1/2} \tag{4}$$

where $(\mathbf{E}_\mathbf{Z})_{i,j} = k(\mathbf{x}_i, \mathbf{z}_j)$ and $(\mathbf{W}_\mathbf{Z})_{i,j} = k(\mathbf{z}_i, \mathbf{z}_j)$. As it can be observed in the above expression, only kernel matrices $\mathbf{W}_\mathbf{Z} \in \mathbb{R}^{r\times r}$ and $\mathbf{E}_\mathbf{Z} \in \mathbb{R}^{n\times r}$ need to be constructed, which are computationally efficient since $r \ll n$. Note that, comparing (3) and (4), $\mathbf{L}_\mathbf{Z}$ is directly related to $\mathbf{L}$ by a linear transformation when $r = s$, since

$$\mathbf{W}_\mathbf{Z}^{-1/2} = \mathbf{U}_\mathbf{Z}\Lambda_\mathbf{Z}^{-1/2}\mathbf{U}_\mathbf{Z}^T \tag{5}$$

where $\mathbf{U}_\mathbf{Z}$ and $\Lambda_\mathbf{Z}$ are eigenvectors and the associated eigenvalues of $\mathbf{W}_\mathbf{Z}$, respectively.

Similar to the previous case, we obtain an ensemble of subspace approximations by repeating the landmark selection process with different clustering techniques: the k-means, k-medians, k-medoids, agglomerative clustering [53], and spectral clustering based on k nearest neighbors [54]. Note that, additional clustering algorithms or a single clustering algorithm with different parameterizations can be utilized as well. For algorithms which only perform partitioning and do not provide cluster centroids (e.g., spectral clustering), we calculate the centroid of a cluster as the mean of the features in that cluster. In summary, based on the P different landmark matrices $\mathbf{Z}_1, \ldots, \mathbf{Z}_P$, we obtain P different embeddings $\mathbf{L}_1, \ldots, \mathbf{L}_P$ for the feature set using (4).

B. Representation Learning

Given the kernel-specific dense embeddings, we perform representation learning for each embedding using a multilayer fully connected network to facilitate the design of a task-specific latent space. Note that, though strategies for sharing weights across the different dense embeddings can be employed, in our implementation we make the networks independent. Following the common practice in deep learning systems, at each hidden layer, dropout regularization [6] is used to prevent overfitting and batch normalization [5] is adopted to accelerate training.

C. Fusion Layer With Kernel Dropout

The fusion layer receives the latent representations for each of the RKHS subspace mappings and can admit a variety of fusion strategies to obtain the final representation for prediction tasks. Common merging strategies include concatenation, summation, averaging, multiplication etc. The back propagation algorithm can then be used to optimize for both the parameters of the representation learning and those of the fusion layer jointly to improve the classification accuracy. Given the large number of parameters and the richness of different kernel representations, the training process can lead to overfitting. In order to alleviate this, we propose to impose a *kernel dropout* regularization in addition to the activation dropout in the representation learning phase.

In the typical dropout regularization [6] for training large neural networks, neurons are randomly chosen to be removed from the network along with their incoming and outgoing connections. The process can be viewed as sampling from a large set of possible network architectures with shared weights. In our context, given the ensemble of dense embeddings $\mathbf{L}_1, \ldots, \mathbf{L}_P$, an effective regularization mechanism is needed

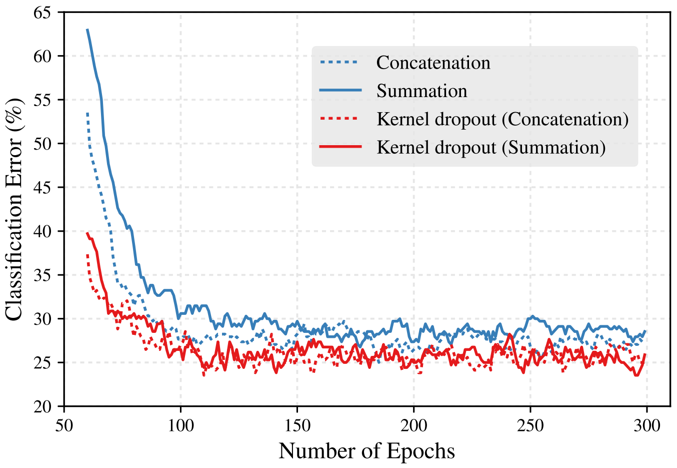


Fig. 2. Effects of kernel dropout on the DKMO training process. We compare the convergence characteristics obtained with the inclusion of the kernel dropout regularization in the fusion layer in comparison to the nonregularized version. Note that we show the results obtained with two different merging strategies—concatenation and summation. We observe that the kernel dropout regularization leads to improved convergence and lower classification error for both the merging styles.

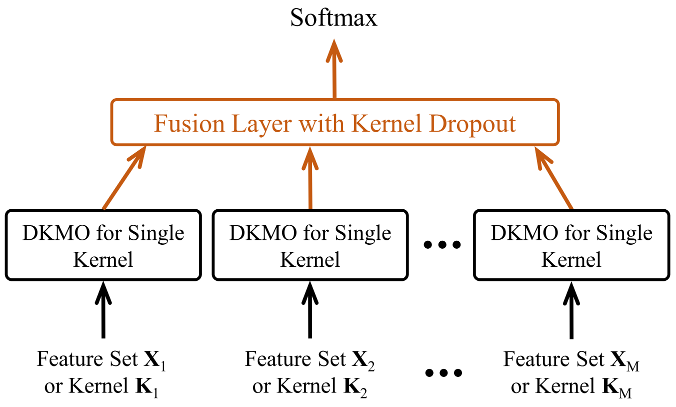

Softmax
Fusion Layer with Kernel Dropout
DKMO for Single Kernel
DKMO for Single Kernel
DKMO for Single Kernel
Feature Set $\mathbf{X}_1$ or Kernel $\mathbf{K}_1$
Feature Set $\mathbf{X}_2$ or Kernel $\mathbf{K}_2$
Feature Set $\mathbf{X}_M$ or Kernel $\mathbf{K}_M$

Fig. 3. M-DKMO—extending the proposed deep kernel optimization approach to the case of multiple kernels. Each of the kernels are first independently trained with the DKMO algorithm in Section III and then combined using a global fusion layer. The parameters of the global fusion layer and the individual DKMO networks are fine tuned in an end-to-end learning fashion.

to prevent the network training from overfitting to certain subspaces in the RKHS. More specifically, we propose to regularize the fusion layer by dropping the entire representations learned from some randomly chosen dense embeddings. Denoting the hidden layer representations before the fusion as $\mathcal{H} = \{\mathbf{h}_p\}_{p=1}^{P}$ and a vector $\mathbf{t}$ associated with P independent Bernoulli trials, the representation $\mathbf{h}_p$ is dropped from the fusion layer if t_p is 0. The feed-forward operation can be expressed as

$$\begin{aligned} t_p &\sim \text{Bernoulli}(P) \\ \tilde{\mathcal{H}} &= \{\mathbf{h} \mid \mathbf{h} \in \mathcal{H} \text{ and } t_p > 0\} \\ \tilde{\mathbf{h}} &= (\mathbf{h}_i), \mathbf{h}_i \in \tilde{\mathcal{H}} \\ \tilde{y}_i &= f(\mathbf{w}_i \tilde{\mathbf{h}} + b_i) \end{aligned}$$

where $\mathbf{w}_i$ are the weights for hidden unit i, $(\cdot)$ denotes vector concatenation and f is the softmax activation function. In Fig. 2, we illustrate the effects of kernel dropout on the convergence speed and classification performance of the network. The results shown are obtained using one of the kernels used in protein subcellular localization (details in Section V-B). We observe that, for both the merging strategies (concatenation and summation), using the proposed regularization leads to improved convergence and produces lower classification error, thereby evidencing improved generalization of kernel machines trained using the proposed approach.

IV. M-DKMO: Extension to Multiple Kernel Learning

As described in Section II-A, extending kernel learning techniques to the case of multiple kernels is crucial to enabling automated kernel selection and fusion of multiple feature sources. The latter is particularly common in complex recognition tasks where the different feature sources characterize distinct aspects of data and contain complementary information. Unlike the traditional kernel construction procedures, the problem of multiple kernel learning is optimized with a task-specific objective, for example hinge loss in classification. In this section, we describe the multiple kernel variant of the DKMO (M-DKMO) presented in the previous section.

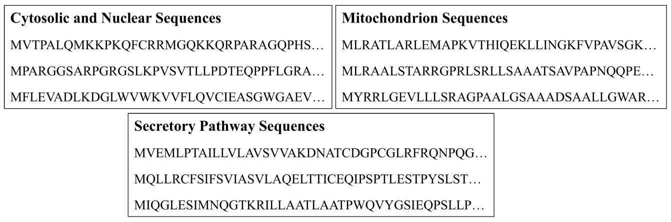
In order to optimize kernel machines with multiple kernels $\{\mathbf{K}\}_{m=1}^{M}$ (optionally feature sets $\{\mathbf{X}\}_{m=1}^{M}$), we begin by employing the DKMO approach to each of the kernels independently. As we will demonstrate with the experimental results, the representations for the individual kernels obtained using the proposed approach produce superior class separation compared to conventional kernel machine optimization (e.g., kernel SVM). Consequently, the hidden representations from the learned networks can be used to subsequently obtain more effective features by exploiting the correlations across multiple kernels. Fig. 3 illustrates the M-DKMO algorithm for multiple kernel learning. As shown in Fig. 3, an end-to-end learning network is constructed based on a set of pretrained DKMO models corresponding to the different kernels and a global fusion layer that combines the hidden features from those networks. Similar to the DKMO architecture in Fig. 1, the global fusion layer can admit any merging strategy and can optionally include additional fully connected layers before the softmax layer.

Note that, after pretraining the DKMO network for each of the kernels with a softmax layer, we ignore the final softmax layer and use the optimized network parameters to initialize the M-DKMO network in Fig. 3. Furthermore, we adopt the kernel dropout strategy described in Section III-C in the global fusion layer before applying the merge strategy. This regularization process guards against overfitting of the predictive model to any specific kernel and provides much improved generalization. From our empirical studies, we observed that both our initialization and regularization strategies enable consistently fast convergence.

V. Experimental Results

In this section, we demonstrate the features and performance of the proposed framework using threefold experiments on real-world data sets:

(a)

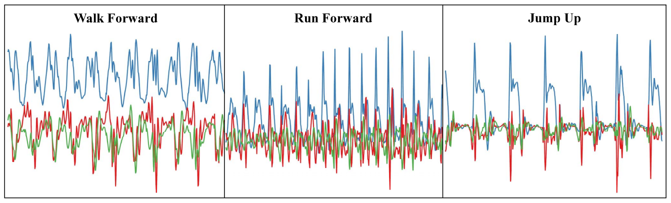


Cytosolic and Nuclear Sequences
MVTPALQMKKPKQFCRRMGQKKQRPARAGQPHS…
MPARGGSARPGRGSLKPVSVTLLPDTEQPPFLGRA…
MFLEVADLKDGLWVWKVVFLQVCIEASGWGAEV…

Mitochondrion Sequences
MLRATLARLEMAPKVTHIQEKLLINGKFVPAVSGK…
MLRAALSTARRGPRLSRLLSAAATSAVPAPNQQPE…
MYRRLGEVLLLSRAGPAALGSAAADSAALLGWAR…

Secretory Pathway Sequences
MVEMLPTAILLVLAVSVVAKDNATCDGPCGLRFRQNPQG…
MQLLRCFSIFSVIASVLAQELTTICEQIPSPTLESTPYSLST…
MIQGLESIMNQGTKRILLAATLAATPWQVYGSIEQPSLLP…

(b)



(c)

Fig. 4. Example samples from the data sets used in our experiments. The feature sources and kernels are designed based on state-of-the-art practices. The varied nature of the data representations are readily handled by the proposed approach and kernel machines are trained for single and multiple kernel cases. (a) Images from different classes in the flowers102 data set. (b) Sequences belonging to three different classes in the nonplant data set for protein subcellular localization. (c) Accelerometer measurements characterizing different activities from the USC-HAD data set.

1) In Section V-A, we compare with single kernel optimization (kernel SVM) and MKL to demonstrate that the proposed methods are advantageous to the existing algorithms based on kernel methods. To this end, we utilize the standard flowers image classification data sets with precomputed features. A sample set of images from this data set are shown in Fig. 4(a).
2) In Section V-B, we emphasize the effectiveness of proposed architectures when only pairwise similarities are available from raw data. In subcellular localization, a typical problem in bioinformatics, the data is in the form of protein sequences [as shown in Fig. 4(b)] and as a result, DNN cannot be directly applied for representation learning. In this experiment, we compare with decomposition-based feature extraction (Decomp) and existing MKL techniques.
3) In Section V-C, we focus on the performance of the proposed architecture when limited training data is available. As a representative application, sensor-based activity recognition requires often laborious data acquisition from human subjects. The difficulty in obtaining large amounts of clean and labeled data can be further complicated by sensor failure, human error and incomplete coverage of the demographic diversity [55]–[57]. Therefore, it is significant to have a model which has strong extrapolation ability given even very limited training data. When features are accessible, an alternative general-purpose algorithm is fully connected neural networks (FCNs) coupled with feature fusion. In this section, we compare the performance of our approach with both FCN and state-of-the-art kernel learning algorithms. A demonstrative set of time-varying measurements are presented in Fig. 4(c).

As can be seen, the underlying data representations considered in our experiments are vastly different, i.e., images, biological sequences, and time series, respectively. The flexibility of the proposed approach enables its use in all these cases without additional preprocessing or architecture fine tuning. Besides, depending on the application we might have access to the different feature sources or to only the kernel similarities. As described in Section III-A, the proposed DKMO algorithm can handle both these scenarios by constructing the dense embeddings suitably.

We summarize all methods used in our comparative studies and the details of the parameters used in our experiments below:

Kernel SVM: A single kernel SVM is applied on each of the kernels. Following [58], the optimal C parameters for kernel SVM were obtained based on a grid search on $[10^{-1}, 10^{0}, 10^{1}, 10^{2}] \times C^*$ through cross validation on the training set, where the default value C^* was calculated as $C^* = 1/((1/n)\sum_i \mathbf{K}_{i,i} - (1/n^2)\sum_{ij} \mathbf{K}_{i,j})$, which is the inverse of the empirical variance of data in the input space.

Uniform: Simple averaging of base kernels has been shown to be a strong baseline in comparison to MKL [17], [18]. We then apply kernel SVM on the averaged kernel.

UFO-MKL: We compare with this state-of-the-art MKL algorithm [59]. The optimal C parameters were cross-validated on the grid $[10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}]$.

Decomp: When only kernel similarities are directly accessible (Section V-B), we compute decomposition-based features using truncated SVD. A linear SVM is then learned on the features with similar parameter selection procedure as in kernel SVM.

Concat: In order to extend Decomp to the multiple kernel case, we concatenate all Decomp features before learning a classifier.

FCN: We construct a fully connected network for each feature set (using Decomp feature if only kernels are available) consisting of four hidden layers with sizes $256-512-256-128$, respectively. For the multiple kernel case, a concatenation layer merges all FCN built on each set. In the training process, batch normalization and dropout with fixed rate of 0.5 are used after every hidden layer. The optimization was carried out using the Adam optimizer, with the learning rate set at 0.001.

DKMO: and *M-DKMO:* For all the data sets, we first applied the DKMO approach to each of the kernels (as in Fig. 1) with the same network size as in FCN: Based on the discussion in Section III-A, for data sets that allow access to explicit feature sources, we extracted five dense embeddings corresponding to

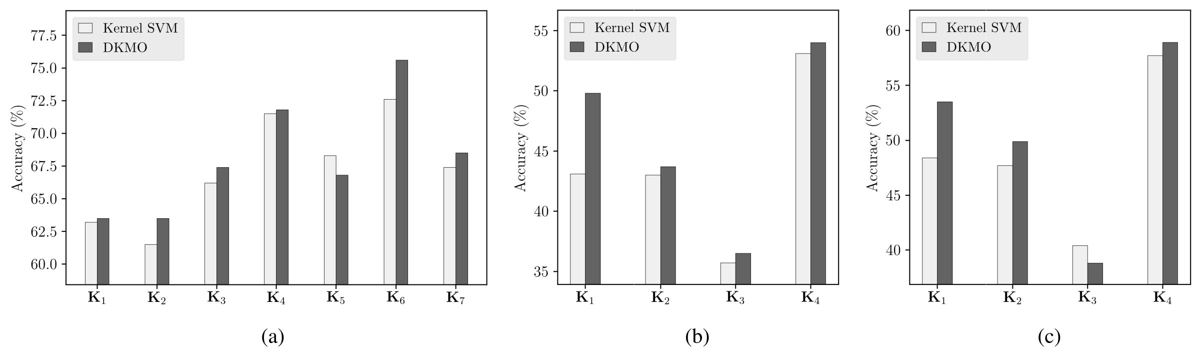


Fig. 5. Single kernel performance on flowers data sets. (a) Flowers17. (b) Flowers102–20. (c) Flowers102–30.

the five landmark point sets obtained using different clustering algorithms. On the other hand, for data sets with only kernel similarity matrices between the samples, we constructed six different dense embeddings with varying subset sizes and approximation ranks. We performed kernel dropout regularization with summation merging for the fusion layer in the DKMO architecture. The kernel dropout rate was fixed at 0.5. For multiple kernel fusion using the M-DKMO approach, we normalize each kernel as $\bar{\mathbf{K}}_{i,j} = \mathbf{K}_{i,j}/(\mathbf{K}_{i,i}\mathbf{K}_{j,j})^{1/2}$, so that $\bar{\mathbf{K}}_{i,i} = 1$. Similar to the DKMO case, we set the kernel dropout rate at 0.5 and used summation based merging at the global fusion layer in M-DKMO. Other network learning parameters were the same as the ones in the FCN method. All network architectures were implemented using the Keras library [60] with the TensorFlow backend and trained on a single GTX 1070 GPU.

A. Image Classification—Comparisons With Kernel Optimization and Multiple Kernel Learning

In this section, we consider the performance of the proposed approach in image classification tasks, using data sets which have had proven success with kernel methods. More specifically, we compare DKMO with kernel SVM, and M-DKMO with Uniform, and UFO-MKL, respectively, to demonstrate that one can achieve better performance by replacing the conventional kernel learning strategies with the proposed deep optimization. We adopt flowers17 and flowers102,[1] two standard benchmarking data sets for image classification with kernel methods. Both data sets are comprised of flower images belonging to 17 and 102 categories, respectively. The precomputed χ^2 distance matrices were calculated based on bag of visual words of features such as HOG, HSV, and SIFT. The variety of attributes enables the evaluation of different fusion algorithms: a large class of features that characterize colors, shapes, and textures can be exploited while discriminating between different image categories [40], [61]–[63].

We construct χ^2 kernels from these distance matrices as $k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma l(\mathbf{x}_i, \mathbf{x}_j)}$, where l denotes the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Following [43], the γ value is empirically estimated as the inverse of the average pairwise distances. To be consistent with the setting from [64] on the flowers102 data set, we consider training on both 20 samples per class and 30 samples per class, respectively. The experimental results for single kernels are shown in Fig. 5 and results for multiple kernel fusion are shown in Table I, where we measure the classification accuracy as the averaged fraction of correctly predicted labels among all classes. As can be seen, DKMO achieves competitive or better accuracy on all single kernel cases and M-DKMO consistently outperforms UFO-MKL. In many cases the improvements are significant, for example, kernel 6 in the flowers17 data set, kernel 1 in the flowers102 data set and the multiple kernel fusion result for the flowers17 data set.

TABLE I
MULTIPLE KERNEL FUSION PERFORMANCE ON FLOWERS DATA SETS

Uniform	UFO-MKL	M-DKMO
FLOWERS17, $n = 1360$		
85.3	87.1	**90.6**
FLOWERS102 - 20, $n = 8189$		
69.9	75.7	**76.5**
FLOWERS102 - 30, $n = 8189$		
73.0	80.4	**80.7**

B. Protein Subcellular Localization—Lack of Explicit Feature Sources

In this section, we consider the case where features are not directly available from data. This is a common scenario for many problems in bioinformatics, where conventional kernel methods have been successfully applied [65]–[67]. More specifically, we focus on predicting the protein subcellular localization from protein sequences. We use 4 data sets from [65][2]: plant, nonplant, psort+, and psort− belonging to $3-5$ classes. Among the 69 sequence motif kernels, we subselect six, which encompass all five patterns for each substring format (except for psort−, where one invalid kernel is removed). Following standard practice, a 50–50 random split is performed to obtain the train and test sets. Since explicit feature sources are not available, the dense embeddings are obtained using the conventional Nyström sampling method.

The experimental results are shown in Fig. 6 and Table II. The first observation is that for Decomp, although the optimal

[1]www.robots.ox.ac.uk/ vgg/data/flowers

[2]www.raetschlab.org/suppl/protsubloc

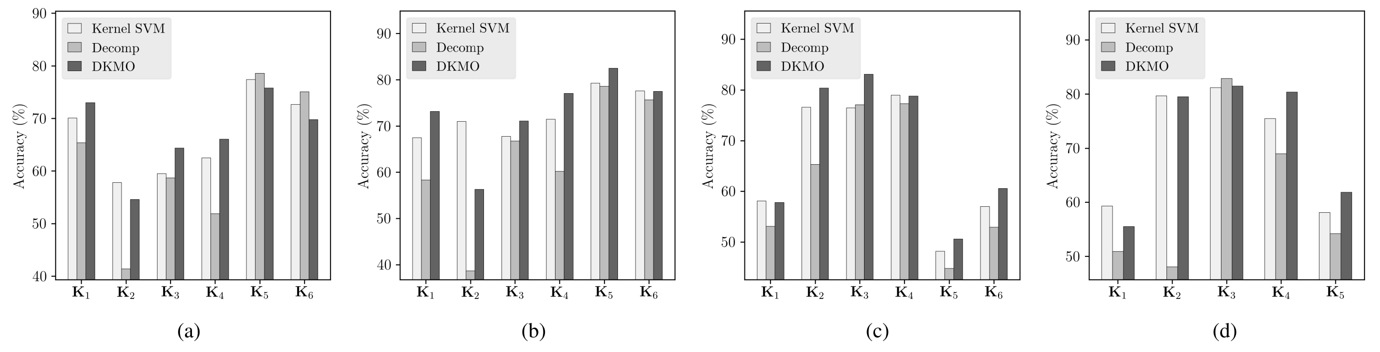


Fig. 6. Single kernel performance on protein subcellular data sets. (a) Plant. (b) Nonplant. (c) Psort+. (d) Psort−.

TABLE II
MULTIPLE KERNEL FUSION PERFORMANCE ON PROTEIN SUBCELLULAR DATA SETS

Concat	**Uniform**	**UFO-MKL**	**M-DKMO**
PLANT, $n = 940$			
90.4	90.3	90.4	**90.9**
NON-PLANT, $n = 2732$			
88.4	91.1	90.3	**93.8**
PSORT+, $n = 541$			
80.6	80.1	**82.8**	82.4
PSORT−, $n = 1444$			
82.5	85.7	**89.1**	87.2

decomposition is used to obtain the features, the results are still inferior and inconsistent. This demonstrates that under such circumstances when features are not accessible, it is necessary to work directly from kernels and build the model. Second, we observe that on all data sets, DKMO consistently produces improved or at least similar classification accuracies in comparison to the baseline kernel SVM. For the few cases where DKMO is inferior, for example, kernel 2 in nonplant, the quality of the Nyström approximation seemed to be the reason. By adopting more sophisticated approximations, or increasing the size of the ensemble, one can possibly make DKMO more effective in such scenarios. Furthermore, in the MKL case, the proposed M-DKMO approach produces improved performance consistently.

Finally, in order to understand the behavior of the representations generated by different approaches, we employ the t-SNE algorithm [68] and obtain 2-D visualizations of the considered baselines and the proposed approaches (Fig. 7). For demonstration, we consider the representation from Decomp of kernel 5 and Decomp of the kernel from Uniform in the nonplant data set. In both DKMO, and M-DKMO, we performed t-SNE on the representation obtained from the fusion layers. The comparisons in Fig. 7 show that the proposed single kernel learning and kernel fusion methods produce highly discriminative representations than the corresponding conventional approaches.

C. Sensor-Based Activity Recognition—Limited Data Case

In this section, we focus on evaluating the performance of the proposed architectures where training data are limited. A typical example under this scenario is sensor-based activity recognition, where the sensor time-series data have to be obtained from human subjects through long-term physical activities. For evaluation, we compare (M)-DKMO with both FCN and kernel learning algorithms.

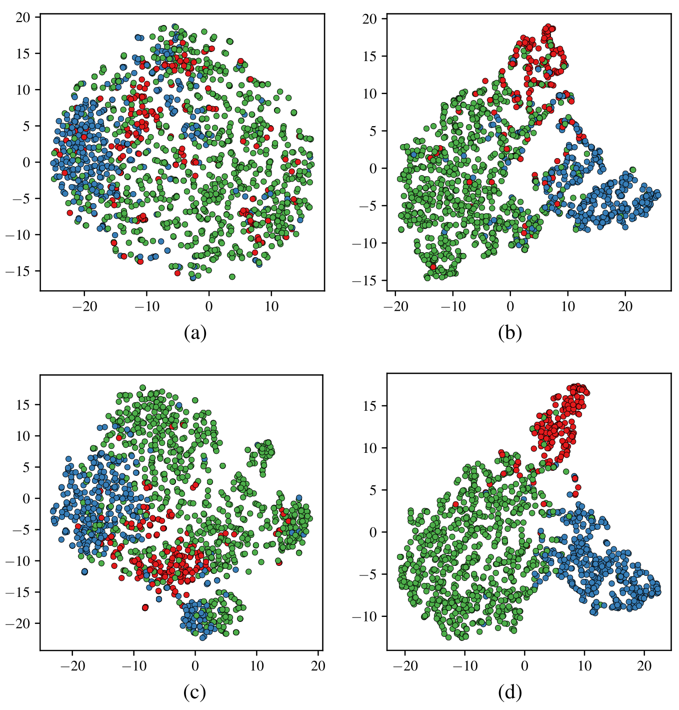


Fig. 7. 2-D T-SNE visualizations of the representations obtained for the nonplant data set using the base kernel (Kernel 5), uniform multiple kernel fusion, and the learned representations from DKMO and M-DKMO. The samples are colored by their corresponding class associations. (a) Decomp. (b) Proposed DKMO. (c) Uniform. (c) Proposed M-DKMO.

Recent advances in activity recognition have demonstrated promising results in fitness monitoring and assisted living [56], [69]. However, when applied to smartphone sensors and wearables, existing algorithms still have limitations dealing with the measurement inaccuracies and noise. Song *et al.* [57] proposed to address this challenge by performing sensor fusion, wherein each sensor is characterized by multiple feature sources, which naturally enables MKL schemes.

We evaluate the performance of our framework using the USC-HAD data set,[3] which contains 12 different daily activities performed by each of the subjects. The measurements

[3]sipi.usc.edu/HAD

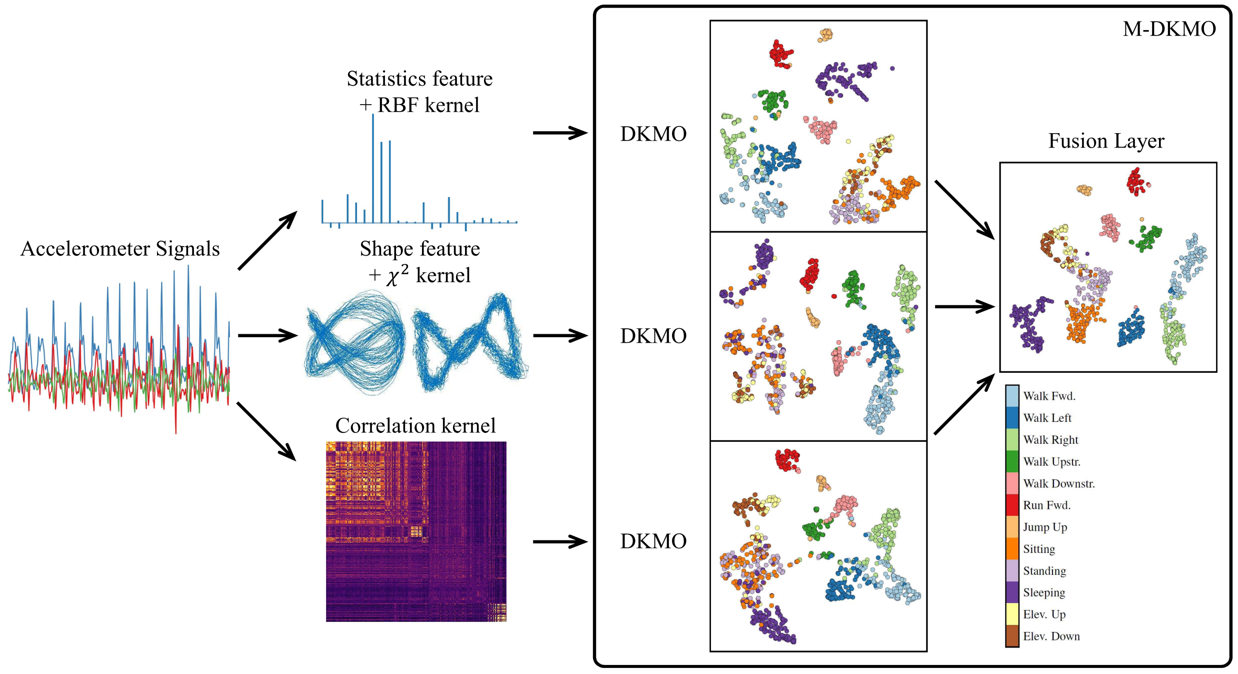


Fig. 8. Visualization of the proposed framework applied on USD-HAD data set. We show the raw three-axis accelerometer signal and extracted three distinct types of features: the time-series statistics, topological structure where we extract TDE descriptors and the correlation kernel. Furthermore, we show the t-SNE visualization of the representations learned by DKMO and M-DKMO, where all points are classes coded according to the colorbar.

are obtained using a 3-axis accelerometer at a sampling rate of 100 Hz. Following the standard experiment methodology, we extract nonoverlapping frames of 5 s each, creating a total of 5353 frames. We perform an 80–20 random split on the data to generate the train and test sets. In order to characterize distinct aspects of the time-series signals, we consider three sets of features:

1) Statistics feature including mean, median, standard deviation, kurtosis, skewness, total acceleration, mean-crossing rate, and dominant frequency. These features encode the statistical characteristics of the signals in both time and frequency domains.
2) Shape feature derived from time-delay embeddings (TDEs) to model the underlying dynamical system [70]. The TDEs of a time-series signal $\mathbf{x}$ can be defined as a matrix $\mathbf{S}$ whose ith row is $\mathbf{s}_i = [x_i, x_{i+\tau}, \ldots, x_{t+(d'-1)\tau}]$, where d' is number of samples and τ is the delay parameter. The time-delayed observation samples can be considered as points in $\mathbb{R}^{d'}$, which is referred as the delay embedding space. In this experiment, the delay parameter τ is fixed to 10 and embedding dimension d' is chosen to be 8. Following the approach in [70], we use principle component analysis to project the embedding to 3-D for noise reduction. To model the topology of the delayed observations in 3-D, we measure the pairwise distances between samples as $\|\mathbf{s}_i - \mathbf{s}_j\|_2$ [71] and build the distance histogram feature with a prespecified bin size.
3) Correlation features characterizing the dependence between time-series signals. We calculate the absolute value of the Pearson correlation coefficient. To account for shift between the two signals, the maximum absolute coefficient for a small range of shift values is identified. We ensure that the correlation matrix is a valid kernel by removing the negative eigenvalues. Given the eigen decomposition of the correlation matrix $\mathbf{R} = \mathbf{U_R}\Lambda_\mathbf{R}\mathbf{U}_\mathbf{R}^T$, where $\Lambda_\mathbf{R} = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ and $\sigma_1 \geq \cdots \geq \sigma_r \geq 0 \geq \sigma_{r+1} \geq \ldots \geq \sigma_n$, the correlation kernel is constructed as $\mathbf{K} = \mathbf{U_R}\hat{\Lambda}_R\mathbf{U}_\mathbf{R}^T$, where $\hat{\Lambda}_R = \operatorname{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$.

Fig. 8 illustrates the overall pipeline of this experiment. As it can be observed, the statistics and shape representations are explicit feature sources and hence the dense embeddings can be constructed using the clustered Nyström method (through RBF and χ^2 kernel formulations, respectively). On the other hand, the correlation representation is obtained directly based on the similarity metric and hence we employ the conventional Nyström approximations on the kernel. However, regardless of the difference in dense embedding construction, the kernel learning procedure is the same for both cases. From the t-SNE visualizations in Fig. 8, we notice that the classes sitting, standing, elevator up, and elevator down are difficult to discriminate using any of the individual kernels. In comparison, the fused representation obtained using the M-DKMO algorithm results in a much improved class separation, thereby demonstrating the effectiveness of the proposed kernel fusion architecture.

From the classification results in Fig. 9, we observe that although FCN obtains better result on the set of statistics features, it has inferior performance on shape and correlation features. On the contrary, DKMO improves on kernel SVM significantly for each individual feature set and is more consistent than FCN. In the case of multiple kernel fusion in Table III, we have striking observations: 1) For FCN,

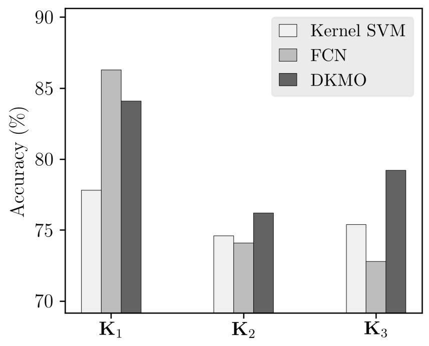


Fig. 9. Single kernel performance on USC-HAD data sets.

TABLE III
MULTIPLE KERNEL FUSION PERFORMANCE ON USC-HAD DATA SETS

Uniform	**UFO-MKL**	**FCN**	**M-DKMO**
USC-HAD, $n = 5353$			
89.0	87.1	85.9	**90.4**

the fusion performance is in fact dragged down by the poor performance on shape and correlation features as in Fig. 9. 2) The uniform merging of kernels is a very strong baseline and the state-of-the-art UFO-MKL achieves lesser performance. 3) The proposed M-DKMO framework further improves over uniform merging, thus evidencing its effectiveness in optimizing with multiple feature sources.

VI. CONCLUSION

In this paper, we presented a novel approach to perform kernel learning using deep architectures. The proposed approach utilizes the similarity kernel matrix to generate an ensemble of dense embeddings for the data samples and employs end-to-end deep learning to infer task-specific representations. Intuitively, we learn representations describing the characteristics of different linear subspaces in the RKHS. By enabling the neural network to exploit the native space of a predefined kernel, we obtain models with much improved generalization. Furthermore, the kernel dropout process allows the predictive model to exploit the complementary nature of the different subspaces and emulate the behavior of kernel fusion using a backpropagation-based optimization setting. In addition to improving upon the strategies adopted in kernel machine optimization, our approach demonstrates improvements over conventional kernel methods in different applications. We also showed that using these improved representations, one can also perform MKL efficiently. In addition to showing good convergence characteristics, the M-DKMO approach consistently outperforms the state-of-the-art MKL methods. The empirical results clearly evidence the usefulness of using deep networks as an alternative approach to building kernel machines. From another viewpoint, similar to the recent approaches such as the convolutional kernel networks [23], principles from kernel learning theory can enable the design of novel training strategies for neural networks. This can be particularly effective in applications that employ fully connected networks and in scenarios where training data is limited, wherein bridging these two paradigms can lead to capacity-controlled modeling for better generalization.

REFERENCES

[1] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.
[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.
[4] S. Abu-El-Haija *et al.* (Sep. 2016). "YouTube-8M: A large-scale video classification benchmark." [Online]. Available: https://arxiv.org/abs/1609.08675
[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[8] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *Int. J. Comput. Vis.*, vol. 73, no. 2, pp. 213–238, Jun. 2007.
[9] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *J. Mach. Learn. Res.*, vol. 2, pp. 419–444, Feb. 2002.
[10] S. V. N. Vishwanathan, N. N. Schraudolph, I. R. Kondor, and K. M. Borgwardt, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, Mar. 2010.
[11] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *J. Mach. Learn. Res.*, vol. 11, pp. 2079–2107, Jul. 2010.
[12] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a Gram matrix for improved kernel-based learning," *J. Mach. Learn. Res.*, vol. 6, pp. 2153–2175, Dec. 2005.
[13] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. NIPS*, 2007, vol. 3. no. 4, pp. 1177–1184.
[14] M. Gönen and E. Alpaydín, "Multiple kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 12, pp. 2211–2268, Jul. 2011.
[15] Z. Sun, N. Ampornpunt, M. Varma, and S. Vishwanathan, "Multiple kernel learning and the SMO algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2361–2369.
[16] J. Li and S. Sun, "Nonlinear combination of multiple kernels for support vector machines," in *Proc. 20th Int. Conf. Pattern Recognit. (ICPR)*, 2010, pp. 2889–2892.
[17] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Oct. 2009, pp. 221–228.
[18] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 396–404.
[19] A. Zien and C. S. Ong, "Multiclass multiple kernel learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 1191–1198.
[20] C. Cortes, M. Mohri, and A. Rostamizadeh, "Multi-class classification with maximum margin multiple kernel," in *Proc. ICML*, 2013, pp. 46–54.
[21] H. Song, J. J. Thiagarajan, P. Sattigeri, K. N. Ramamurthy, and A. Spanias, "A deep learning approach to multiple kernel fusion," in *Proc. IEEE ICASSP*. New Orleans, LA, USA, Mar. 2017, pp. 2292–2296.
[22] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2627–2635.
[23] J. Mairal, "End-to-end kernel learning with supervised convolutional kernel networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1399–1407.
[24] A. M. Andrew, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, N. Christianini and J. Shawetaylor, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2000, p. 189.
[25] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 349–366, Feb. 2007.

[26] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: Spectral clustering and normalized cuts," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2004, pp. 551–556.

[27] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.

[28] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Multiple kernel sparse representations for supervised and unsupervised learning," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 2905–2915, Jul. 2014.

[29] J. J. Thiagarajan, K. Ramamurthy, A. Spanias, and D. Frakes, "Kernel sparse models for automated tumor segmentation," U.S. Patent 9 710 916, Jul. 18, 2017.

[30] M. Harandi, M. Salzmann, S. Jayasumana, R. Hartley, and H. Li, "Expanding the family of Grassmannian kernels: An embedding perspective," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 408–423.

[31] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the Nyström method," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 981–1006, 2012.

[32] K. Zhang and J. T. Kwok, "Clustered Nyström method for large scale manifold learning and dimension reduction," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1576–1587, Oct. 2010.

[33] S. Kumar, M. Mohri, and A. Talwalkar, "Ensemble nystrom method," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1060–1068.

[34] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1177–1184.

[35] Q. Le, T. Sarlos, and A. Smola, "Fastfood—Approximating kernel expansions in loglinear time," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, 2013, pp. 244–252. [Online]. Available: http://jmlr.org/proceedings/papers/v28/le13.html

[36] P. S. Huang, H. Avron, T. N. Sainath, V. Sindhwani, and B. Ramabhadran, "Kernel methods match deep neural networks on timit," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 205–209.

[37] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *J. Mach. Learn. Res.*, vol. 9, pp. 2491–2521, Nov. 2008.

[38] A. Jain, S. V. Vishwanathan, and M. Varma, "SPG-GMKL: Generalized multiple kernel learning with a million kernels," in *Proc. 18th KDD*, 2012, pp. 750–758.

[39] P. Natarajan *et al.*, "Multimodal feature fusion for robust event detection in Web videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 1298–1305.

[40] S. S. Bucak, R. Jin, and A. K. Jain, "Multiple kernel learning for visual object recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1354–1369, Jul. 2014.

[41] F. Liu, L. Zhou, C. Shen, and J. Yin, "Multiple kernel learning in the primal for multimodal alzheimer's disease classification," *IEEE J. Biomed. Health Inform.*, vol. 18, no. 3, pp. 984–990, May 2014.

[42] H. Song, J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Auto-context modeling using multiple kernel learning," in *Proc. IEEE ICIP*, Phoenix, AZ, USA, Sep. 2016, pp. 1868–1872.

[43] F. Orabona, L. Jie, and B. Caputo, "Multi kernel learning with online-batch optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 227–253, Feb. 2012.

[44] M. Gönen and E. Alpaydin, "Localized multiple kernel learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 352–359.

[45] R. Kannao and P. Guha, "TV commercial detection using success based locally weighted kernel combination," in *MultiMedia Modeling*. Berlin, Germany: Springer, 2016, pp. 793–805.

[46] J. Moeller, S. Swaminathan, and S. Venkatasubramanian, "A unified view of localized kernel learning," in *Proc. SIAM Int. Conf. Data Mining*, 2016, pp. 252–260.

[47] J. Zhuang, I. W. Tsang, and S. C. Hoi, "Two-layer multiple kernel learning," in *Proc. AISTATS*, 2011, pp. 909–917.

[48] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 342–350.

[49] M. A. Wiering and L. R. Schomaker, "Multi-layer support vector machines," in *Regularization, Optimization, Kernels, and Support Vector Machines*. Boca Raton, FL, USA: CRC Press, 2014, pp. 457–475.

[50] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Proc. 19th Int. Conf. Artif. Intell. Stat.*, 2016, pp. 370–378.

[51] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2177–2185.

[52] M. Li, W. Bi, J. T. Kwok, and B.-L. Lu, "Large-scale Nyström kernel matrix approximation using randomized SVD," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 152–164, Jan. 2015.

[53] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, vol. 344. Hoboken, NJ, USA: Wiley, 2009.

[54] U. von Luxburg, "A tutorial on spectral clustering," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[55] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newslett.*, vol. 12, no. 2, pp. 74–82, Dec. 2010.

[56] M. Zhang and A. A. Sawchuk, "USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 1036–1043.

[57] H. Song, J. J. Thiagarajan, K. N. Ramamurthy, A. Spanias, and P. Turaga, "Consensus inference on mobile phone sensors for activity recognition," in *Proc. IEEE ICASSP*, Shanghai, China, Mar. 2016, pp. 2294–2298.

[58] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. AISTATS*, 2005, pp. 57–64.

[59] F. Orabona and L. Jie, "Ultra-fast optimization algorithm for sparse multi kernel learning," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 249–256.

[60] F. Chollet *et al.* (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras

[61] I.-H. Jhuo and D. Lee, "Boosted multiple kernel learning for scene category recognition," in *Proc. 20th Int. Conf. Pattern Recognit. (ICPR)*, 2010, pp. 3504–3507.

[62] J. J. Thiagarajan, K. N. Ramamurthy, P. Turaga, and A. Spanias, "Image understanding using sparse representations," *Synth. Lectures Image, Video, Multimedia Process.*, vol. 7, no. 1, pp. 1–118, 2014.

[63] K. Ramamurthy, J. J. Thiagarajan, P. Sattigeri, and A. Spanias, "Ensemble sparse models for image analysis and restoration," U.S. Patent 14 772 343, Jan. 14, 2016.

[64] X. Qi, R. Xiao, C.-C. Li, Y. Qiao, J. Guo, and X. Tang, "Pairwise rotation invariant co-occurrence local binary pattern," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2199–2213, Nov. 2014.

[65] C. S. Ong and A. Zien, "An automated combination of kernels for predicting protein subcellular localization," in *Proc. Int. Workshop Algorithms Bioinformatics*, 2008, pp. 186–197.

[66] C. H. Ding and I. Dubchak, "Multi-class protein fold recognition using support vector machines and neural networks," *Bioinformatics*, vol. 17, no. 4, pp. 349–358, 2001.

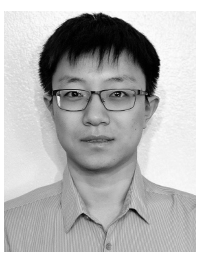
[67] A. Andreeva, D. Howorth, C. Chothia, E. Kulesha, and A. G. Murzin, "SCOP2 prototype: A new approach to protein structure mining," *Nucl. Acids Res.*, vol. 42, no. D1, pp. D310–D314, 2014.

[68] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[69] S. Zhang, C. Tepedelenlioğlu, M. K. Banavar, and A. Spanias, "Max consensus in sensor networks: Non-linear bounded transmission and additive noise," *IEEE Sensors J.*, vol. 16, no. 24, pp. 9089–9098, Dec. 2016.

[70] J. Frank, S. Mannor, and D. Precup, "Activity and gait recognition with time-delay embeddings," in *Proc. AAAI*, 2010, pp. 1581–1586.

[71] V. Venkataraman and P. Turaga, "Shape distributions of nonlinear dynamical systems for video-based inference," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 12, pp. 2531–2543, Dec. 2016.



Huan Song (M'18) received the B.S. degree in electrical engineering from Xidian University, Xi'an, China, in 2012, and the M.S. degree in electrical engineering from Arizona State University (ASU), Tempe, AZ, USA, in 2015, where he is currently pursuing the Ph.D. degree with the School of Electrical, Computer, and Energy Engineering.

His current research interests include multimodal fusion methods utilizing deep learning and multiple kernel methods for time-series analysis and graph mining.

Jayaraman J. Thiagarajan (M'18) received the Ph.D. and M.S. degrees in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2008 and 2013, respectively.

He is currently a Computer Scientist with the Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, USA. He has co-authored two books. His current research interests include machine learning, computer vision, natural language processing, signal processing, topological data analysis, the investigation of using machine learning for science, clinical data analysis, high-performance computing, computed tomography, and building tools for model interpretability.

Dr. Thiagarajan was a recipient of the multiple Best Paper Nominations at premier IEEE conferences. He has served as a reviewer for several IEEE, ACM, Elsevier, and Springer journals and conferences.

Prasanna Sattigeri (M'18) received the Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2015.

He is currently a Research Staff Member with the IBM Research AI, Yorktown Heights, NY, USA. His broad research interests include developing efficient systems for machine learning and semantic inferences from data. His current research interests include deep generative models, learning with limited data, transfer learning and interpretability. He is also interested in developing scalable solutions and has experience shipping machine learning products to large consumer bases.

Andreas Spanias (F'18) is currently a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ, USA, where he is also the Director of the Sensor Signal and Information Processing (SenSIP) Center and the Founder of the SenSIP industry consortium (now an NSF I/UCRC site). He and his student team developed the computer simulation software Java-DSP and its award-winning iPhone/iPad and Android versions. He has authored two text books: *Audio Processing and Coding* (Wiley) and DSP and *An Interactive Approach* (2nd Ed.). His current research interests include adaptive signal processing, speech processing, and sensor systems.

Dr. Spanias was a co-recipient of the 2002 IEEE Donald G. Fink Paper Prize Award. He served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the General Co-Chair of IEEE ICASSP-99. He also served as the IEEE Signal Processing Vice President for Conferences. He served as the Distinguished Lecturer for the IEEE Signal processing society in 2004. He is a Series Editor for the Morgan and Claypool lecture series on algorithms and software.