# Data-Driven Representation Learning in Multimodal Feature Fusion

by

Huan Song

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved June 2018 by the
Graduate Supervisory Committee:

Andreas Spanias, Chair
Jayaraman Thiagarajan
Visar Berisha
Cihan Tepedelenlioglu

ARIZONA STATE UNIVERSITY

August 2018

ABSTRACT

Modern machine learning systems leverage data and features from multiple modalities to gain more predictive power. In most scenarios, the modalities are vastly different and the acquired data are heterogeneous in nature. Consequently, building highly effective fusion algorithms is at the core to achieve improved model robustness and inferencing performance. This dissertation focuses on the representation learning approaches as the fusion strategy. Specifically, the objective is to learn the shared latent representation which jointly exploit the structural information encoded in all modalities, such that a straightforward learning model can be adopted to obtain the prediction.

We first consider sensor fusion, a typical multimodal fusion problem critical to building a pervasive computing platform. A systematic fusion technique is described to support both multiple sensors and descriptors for activity recognition. Targeted to learn the optimal combination of kernels, Multiple Kernel Learning (MKL) algorithms have been successfully applied to numerous fusion problems in computer vision etc. Utilizing the MKL formulation, next we describe an auto-context algorithm for learning image context via the fusion with low-level descriptors. Furthermore, a principled fusion algorithm using deep learning to optimize kernel machines is developed. By bridging deep architectures with kernel optimization, this approach leverages the benefits of both paradigms and is applied to a wide variety of fusion problems.

In many real-world applications, the modalities exhibit highly specific data structures, such as time sequences and graphs, and consequently, special design of the learning architecture is needed. In order to improve the temporal modeling for multivariate sequences, we developed two architectures centered around attention models. A novel clinical time series analysis model is proposed for several critical

problems in healthcare. Another model coupled with triplet ranking loss as metric learning framework is described to better solve speaker diarization. Compared to state-of-the-art recurrent networks, these attention-based multivariate analysis tools achieve improved performance while having a lower computational complexity. Finally, in order to perform community detection on multilayer graphs, a fusion algorithm is described to derive node embedding from word embedding techniques and also exploit the complementary relational information contained in each layer of the graph.

*To*

*my dearest family*

*who made it possible for me to complete this work.*

ACKNOWLEDGEMENTS

First and foremost I wish to express my sincere gratitude to my advisor Professor Andreas Spanias, for continuously supporting my Ph.D study and encouraging my research. He nurtured my interest toward the Ph.D program and gave me the freedom to venture into exciting research topics. His confidence and guidance for me have been invaluable at all the time.

I would like to convey special thank to Dr. Jayaraman Thiagarajan at Lawrence Livermore National Labs, for his tremendous guidance to my research. Many topics in our collaboration are fundamental to this thesis. I am sincerely grateful for the enthusiasm, encouragement and unconditional support he brought to me throughout the collaboration.

I am very thankful to the rest of my thesis committee: Dr. Cihan Tepedelenlioglu and Dr. Visar Berisha, for their valuable comments and advices on this thesis. My collaboration with Dr. Visar Berish have been very exciting and it significantly broadens my research. I would like to extend my gratitude to Dr. Karthikeyan Natesan Ramamurthy, Dr. Prasanna Sattigeri and Ms. Deepta Rajan at IBM Research for the collaboration on various perspectives, which inspired and motivated new research ideas for me. I sincerely appreciate the help from the department staff, in particular, Robina, Sno, Esther, Toni, Donna, Ginger and Jenna, over the years. I thank all former and current fellow labmates at SenSIP Center, Sai, Jongmin, Henry, Uday, Sunil, Jie, Abhinav, Sameeksha, David, Gowtham and Vivek, for the great time we have shared.

Last but not the least, the deepest gratitude to my family for their support and sacrifices. Without their unconditional love, this research would have never been

possible. I am deeply grateful for their encouragement to me to pursue my enthusiasm and for their help to let me through all difficult times.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

# Chapter 1

# INTRODUCTION

## 1.1 Multimodal Feature Fusion

In complex machine learning systems, data often originate from multiple disparate modalities. The modalities can correspond to sensory channels such as images and audio, multiple data sources such as activities on a social network and online purchase history, or distinct descriptors derived from the same input source (e.g. spatial and temporal descriptors on the same video clip). Due to significant variabilities in format, structure, and complexity of the different modalities, machine learning in such scenarios is very challenging, and hence naïve extensions of existing techniques for single source analysis often lead to inferior results with multiple data sources. For example, a simple strategy of handling multiple feature sets by concatenating them will result in very high-dimensional representations that might suffer from the *curse of dimensionality* and incur significantly higher computational complexity. Moreover, the large discrepancy in the statistical characteristics of the different feature sources are usually difficult to exploit for modeling techniques. In light of the above issues, in multimodal learning, it is critical to develop a highly effective fusion model to jointly consider the information contained in all modalities.

In this dissertation, the objective of multimodal feature fusion is to improve the predictive power of the overall system, while comprehensively exploring the power of each of the constituent modalities. The benefit of such fusion methodologies can be attributed to the complementary or supplemental information provided by different modalities. For example, in modern smartphones and wearables, a complex sensing

system consists of abundant sensors: a compass, an accelerometer, a gyroscope and a microphone, to name a few. Utilizing the heterogeneous signals from these modalities can compensate for inherent limitations of such low-cost sensors, including large measurement uncertainty and low signal-to-noise ratio (K. Liu et al. 2014; Gravina et al. 2017). Such systems have been shown to be more robust and can achieve much better accuracy when applied in activity recognition (Zhu and Sheng 2009) and health monitoring (Orwat, Graefe, and Faulwasser 2008) applications. In speech recognition, most existing systems perform modeling based solely on the audio signal. However, the associated visual modality, when available, can provide the extra information to help discriminate acoustics that are ambiguous to audio features. By coupling with the lip movement from the videos, a multimodal neural network which learns a shared representation can provide improved speech understanding (Ngiam et al. 2011).

In other scenarios where distinct descriptors of the data are utilized for fusion, the gained performance improvement can result from combining the discriminative power of different features when distinguishing certain examples. For instance, in the classical bag-of-words approaches for visual recognition, more weighting on the texture feature can help recognize crocodile images while the color feature could be more critical for classifying the strawberry class (Bucak, Jin, and Jain 2014). Finally, for both conventional Support Vector Machine classifier and the more recent deep neural networks, it is observed that incorporating middle- and high-level abstractions including image context descriptors is beneficial to image recognition and segmentation tasks (Tu 2008; Park et al. 2016; Srinivas et al. 2016). In general, multiple descriptors usually characterize salient aspects of the data and the effective fusion can boost the model's capability for discriminating a large number of classes, while being robust to variations within a class.

## 1.2 Why Learn Representations?

Broadly speaking, there are two classes of approaches to accomplish the fusion of multimodal features. The first one is based on decision fusion, where each feature set is used to train an independent model (e.g. classifier/regressor) and the decisions from the models are aggregated using strategies such as majority voting. However, more sophisticated techniques have been developed for decision fusion namely fuzzy set theory (Hong and Choi 2000) and ensemble learning (Polikar 2006). In contrast, the second class of approaches considers the commonalities and differences among the modalities and builds a shared representation prior to the learning stage. Compared to decision fusion, the representation learning strategy provides a unified feature set that can be subsequently used with any chosen learning algorithm. This enables convenient model selection and easy adoption to a variety of tasks. Furthermore, thanks to the concise fused representation, inference can usually be carried out very efficiently.

A perhaps more significant advantage of representation learning based fusion is its ability to perform end-to-end learning. Powered by the advances in deep neural network optimization, this approach infers shared representations that draw consensus across modalities, while simultaneously solving the task at hand. Until recently, expert-designed, hand-engineered features have been prevalent in machine learning community, for example scale-invariant feature transform (SIFT) (Lowe 1999) and histogram of oriented gradients (HOG) (Dalal and Triggs 2005) for computer vision and Mel-frequency cepstral coefficients (MFCC) for speech recognition and audio analysis. However, despite the sophisticated design, hand-engineered features rely almost entirely on prior knowledge on the degrees of freedom required to solve a complex task, and can be grossly insufficient to automatically infers factors of influence from large data corpora. The recent surge in representation learning provides

a data-driven way to extract effective features and has revolutionized machine learning and data analysis. In particular, the success of Deep Neural Networks (DNNs) in a wide variety of computer vision tasks has emphasized the need of representation learning (Nair and Hinton 2010; He et al. 2016) and end-to-end learning. By coupling modern deep architectures with large datasets (Deng et al. 2009; Abu-El-Haija et al. 2016), efficient optimization strategies (Ioffe and Szegedy 2015; Srivastava et al. 2014) and advanced hardware architectures, one can obtain multi-level abstractions from data and highly effective predictive models with unprecedented success.

Due to the aforementioned advantages, in this dissertation, we focus on representation learning strategies for multimodal feature fusion. Nevertheless, it is important to note that in real-world, the modalities can exhibit vastly different structures and specific design of the fusion model is needed to achieve the desired performance. For example, the generic fully connected neural network is widely utilized in autoencoder (Hinton and Salakhutdinov 2006), but usually is not preferred in image modeling and speech analysis tasks. Moreover, in relational data such as graphs, most existing DNN architectures are not directly applicable. In the rest of the dissertation, we discuss the fusion problem in broad domains ranging from time sequences, images to speech and graphs and provide a suite of specific solutions.

## 1.3 Problem Statement

The problem of dealing with multi-modal or more generally multi-faceted data is ubiquitous in several application domains. Furthermore, in each specific scenario, there is often a critical need to adhere to the additional constraints (e.g. limited availability of labeled data) or to exploit prior knowledge about the structure in data (e.g. temporal and relational structure). This dissertation focuses on building mathematical formulations and algorithms to solve the multimodal feature fusion

problem under these scenarios for applications including mobile activity analysis, object recognition, clinical modeling, speech processing and network data analysis.

### 1.3.1 Multimodal Sensor Fusion and Ensemble Inference

The pervasive use of wearable sensors in activity and health monitoring presents a huge potential for building novel multimodal fusion frameworks. In particular, approaches that can harness data from a diverse set of low-cost sensors for recognition are needed. Some of the commonly used sensors include the accelerometer, gyroscope and magnetometer, to name a few. Though abundantly available, the data collected from these cheap sensors are often very noisy and unreliable. Note that, the noise and inaccuracies can be caused by the sensors themselves or due to arbitrary position/movement of the devices during the activities. In the existing literature, in spite of the availability of different sensors, activity recognition is often carried out solely based on accelerometer data (Kwapisz, Weiss, and Moore 2011). An effective fusion mechanism taking into account all modalities will be able to reduce the measurement uncertainty and provide improved robustness, especially in the case of missing data or modalities. This problem assumes access to multiple sensors, and multiple feature descriptors for each of the sensors. Hence, the goal is to automatically fuse this information with the overall objective of improving activity recognition performance.

### 1.3.2 Multiple Kernel Fusion for Visual Context Modeling

In image understanding, it is common practice to fuse features which describe different aspects of objects including shape, color etc. A common characteristic of several existing multimodal feature fusion algorithms is that features employed are often low-level in nature i.e., they describe the local variabilities without taking the global context into account. However, it is known that high-level information,

referred to as the context, is crucial to object/scene understanding (Thiagarajan et al. 2014). Consequently, *auto-context* models (Tu 2008) have been developed, which can approximate the posterior using an iterative, supervisory approach. More specifically, these models integrate the low-level features with context information in the form of probability maps, obtained using a series of classifiers. By enabling the classifier to choose different supporting neighbors to modify the current probabilities towards the ground truth, auto-context methods lead to better regularization. In the original auto-context model, however, the combination of low-level image descriptors and context feature is carried out by simple concatenation, which is insufficient when each modality is non-linearly separable. The goal of this problem is to develop a mathematical formalism that will support fusion of low-level feature descriptors and high-level context characteristics, for example class assignment likelihoods in multi-class classification problems.

### 1.3.3 Deep Kernel Machine Optimization

As described earlier, enabling data fusion with additional constraints or assumptions on the data domain is essential to employing fusion techniques in real-world applications. Prior art in machine learning offers a multitude of solutions to deal with different challenges, however rarely unified solutions exist that can be broadly applicable to such constrained scenarios. In particular, as two significant learning paradigms in machine learning, kernel methods and deep learning have each achieved huge success across many domains in different periods of time. 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 sub-optimal solutions

or model overfitting. This is particularly more common in applications characterized by limited dataset sizes and complex dependencies in the input space. Despite the advances in regularization techniques and data augmentation strategies (Krizhevsky, Sutskever, and Hinton 2012), 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, non-linear predictive models is to employ kernel machines. In recent years, there is increased research interest to combine the advantages of the two methods. Broadly speaking, the existing approaches either utilize kernel compositions to emulate neural network layer stacking or facilitate the optimization of deep architectures with data-specific kernels. Combining the advantages of these two paradigms of predictive learning can potentially lead to new architectures and inference strategies.

### 1.3.4 Multivariate Sequence Modeling using Attention

The classical approach for sequential data analysis has been centered around extracting hand-engineered features and building task-specific predictive models. These models are often challenged by factors such as need for long-term dependencies, irregular sampling and missing values. In the recent years, recurrent Neural Networks (RNNs) based on Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) have become the de facto solution to deal with multivariate sequence data. RNNs are designed to model varying-length data and have achieved state-of-the-art results in sequence-to-sequence modeling (Sutskever, Vinyals, and Le 2014), image captioning (Xu et al. 2015) and recently in clinical diagnosis (Lipton et al. 2015). Furthermore, LSTMs are effective in exploiting long-range dependencies and handling nonlinear dynamics. These recurrent architectures perform computations at each position of the time sequence by generating a series of hidden states as a function of

the previous hidden state and the input for current position. Such inherent sequential nature makes parallelization challenging. Though efforts to improve the computational efficiency of sequential modeling have recently surfaced, some of the limitations still persist. The recent work of Vaswani *et. al.* (Vaswani et al. 2017) argues that attention mechanisms, without any recurrence, can be effective in sequence-to-sequence modeling tasks. Attention mechanisms are used to model dependencies in sequences without regard for their actual distances in the sequence (Bahdanau, Cho, and Bengio 2014). This characteristic is particularly suited for multivariate time sequence modeling, where the complexity lies in both temporal modeling and fusion of the multi-variate measurements. We consider two important problems in multivariate sequence modeling where attention model can be applied to obtain improved sequence modeling: clinical predictions with health records and unsupervised speaker diarization.

### 1.3.5 Node Embedding on Multilayer Graphs for Unsupervised Inference

Finally, exploiting prior knowledge about the structure in data, often represented as geometric objects such as graphs or manifolds, while performing data fusion will enable generalization of existing techniques to novel applications such as social network modeling or brain network analysis. More specifically, using graphs to represent and perform inferencing with relational data has become ubiquitous in data mining and machine learning applications. We focus on the application of graphs for unsupervised learning, in particular the graph mining task of clustering vertices into homogeneous groups. Recently, there has been particular interest in adopting neural word embeddings, which encode semantic relations in a continuous vector space with a relatively small number of dimensions, to representation learning on graphs (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016). At its core, word embedding algorithms build on the distributional hypothesis (Harris 1954) where similar contexts

imply similar meanings, i.e., co-occurrences of words can be tied to their underlying meanings. In real-world application, however, multiple relational information often exist among the same set of graph vertices and such data structure is usually named multilayer graphs. Originally proposed for representation learning, the graph node embedding techniques are not directly application on multilayer graphs. We are interested in extending node embedding to multilayer graphs in order to obtain fused multimodal representations of the graph nodes. To achieve this goal, the intralayer and interlayer node relations both have to be taken into account in the representation learning process.

## 1.4 Contributions

In Chapter 3, a novel two-stage architecture for sensor fusion in activity recognition is described. The algorithm supports data from multiple sensors on the same device and also multiple feature extraction strategies on the same sensor. As mentioned in Section 1.1, the straightforward fusion approach of simply concatenating feature vectors derived from different sensors, for example in (Zhang and A. A Sawchuk 2013), is known not to exploit the inherent geometry of the different feature domains. In our architecture, the first stage performs sensor fusion, for each feature independently, using a linearized variant of the multilayer graph consensus approach (K.N. Ramamurthy et al. 2014). On the second stage, the two sets of consensus features are used to build a reference-based ensemble classifier to make the final prediction. We tested the proposed approach with real data collected from 32 subjects performing primitive activities, and results show that the proposed two stage approach can improve the performance significantly, when compared to using a single sensor. Methods and results of the architecture described in this chapter have been published in (Song, Thiagarajan, Ramamurthy, Spanias, and Turaga 2016).

In Chapter 4, the idea of using multiple kernel learning to perform multimodal fusion is introduced and an improved auto-context model under the RKHS setting is developed for image classification. In addition to providing the flexibility of auto-context models, the proposed approach can build highly effective kernel models for object recognition. Since auto-context probability maps cannot be directly incorporated into the RKHS, we first estimate marginal probabilities using a classifier (e.g. Kernel Logistic Regression or Kernel SVM) and construct an *auto-context* kernel based on these probabilities. For example, the marginalized kernel construction in (Tsuda, Kin, and Asai 2002) can be used. Since any symmetric positive definite kernel defines a unique RKHS, we can use other forms of kernels by treating the probability map for each image as a feature vector directly. Interestingly, the process of fusing the auto-context model with the image appearance can be viewed as Multiple Kernel Learning (see Section 2.2), for which a variety of efficient solutions exist. We demonstrate using standard object/scene classification datasets that the proposed approach results in highly effective recognition systems. The algorithm and experiments reported in this chapter can be found in (Song, Thiagarajan, Ramamurthy, and Spanias 2016).

Continuing the discussion on using kernel methods for multimodal fusion, in Chapter 5, we utilize deep neural networks to enable end-to-end optimization of multiple kernel fusion. The end-to-end *Deep Kernel Machine Optimization* (DKMO) approach builds dense embeddings for data using kernel approximation, learns concise representations and finally infers the predictive model for a given kernel. While existing kernel approximation techniques make kernel learning efficient, utilizing deep networks enables end-to-end inference with a task-specific objective. The extension of DKMO to multiple kernels for multimodal fusion is straightforward and we show that, with appropriate regularization, the proposed multiple kernel learning optimization

is highly efficient, in terms of convergence characteristics. In contrast to approaches such as (Mairal 2016; Mairal et al. 2014), 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 employ fully connected neural networks to infer the predictive model. Consequently, our approach is generic and not restricted to applications that can use only convolutional neural networks. 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. Based on empirical studies with a variety of datasets from cell biology, image classification and activity recognition, we demonstrate the superiority of the proposed approaches in comparison to the baseline kernel SVMs and the state-of-the-art MKL algorithms. This work have been published in (Song et al. 2017; Song, Thiagarajan, Sattigeri, and Spanias 2018).

In Chapter 6, we develop improved multivariate time sequence modeling architectures based on the recent work of attention models. Specifically, we focus on the clinical time series analysis and speaker diarization problems. For the first task, we develop the *SAnD* (Simply Attend and Diagnose) architecture, which employs a masked, self-attention mechanism, and uses positional encoding and dense interpolation strategies for incorporating temporal order. Furthermore, we develop a multi-task variant of *SAnD* to jointly infer models with multiple diagnosis tasks. Using the recent MIMIC-III benchmark datasets, we demonstrate that the proposed approach achieves state-of-the-art performance in all tasks, outperforming LSTM models and classical baselines with hand-engineered features. Regarding the speaker diarization problem, we investigate the importance of learning effective representations using attentions directly under the metric learning pipelines for speaker diarization. More specifically,

we propose to employ attention models to learn embeddings and the metric jointly in an end-to-end fashion. Experiments are conducted on the CALLHOME conversational speech corpus. The diarization results demonstrate that, besides providing a unified model, the proposed approach achieves improved performance when compared against existing approaches. The content of this chapter have been described in (Song, Rajan, et al. 2018; Song, Willi, et al. 2018).

In Chapter 7, we utilize the recent advances as representation learning on graphs, and extend it to multilayer graphs for community detection. A straightforward extension could be performing random walks on each of the layers independently and then merging the latent features. However, this approach does not exploit the common factors or regularity in the community structures across the different layers. Furthermore, the commonly adopted late fusion strategies such as concatenation do not compensate for the uncertainties introduced by each of the layers while identifying clusters. In order to circumvent the aforementioned challenges, we propose a deep embedding approach that utilizes both an early fusion strategy based on supra adjacency matrix construction for exploiting shared information across layers and a late fusion strategy that builds a unified representation for nodes in the graph using optimization with a deep clustering objective.

# Chapter 2

# BACKGROUND

## 2.1 Canonical Correlation Analysis

As a traditional feature fusion method, Canonical Correlation Analysis (CCA) analyses relationships between two sets of variables using the cross-covariance. It seeks the two sets of linear projections such that the correlation between the variables are maximized while the projections within each set are uncorrelated. As pointed out in (Baltrušaitis, Ahuja, and Morency 2017), CCA can be regarded as obtaining *coordinated representations*, where representations for each modality are learned separately but are coordinated by the orthogonality constraint.

Given the feature domain $\mathcal{X} \subset \mathbb{R}^d$, we define the two matrices of $n$ samples as $\mathbf{X}_1, \mathbf{X}_2$. Denote the two covariances as $\mathbf{\Sigma}_{11}, \mathbf{\Sigma}_{22}$ and the cross-covariance $\mathbf{\Sigma}_{12}$. CCA finds the linear projections $\mathbf{w}_1\mathbf{X}_1, \mathbf{w}_2\mathbf{X}_2$ which are maximally correlated:

$$\begin{aligned} \mathbf{w}_1^*, \mathbf{w}_2^* &= \arg\max \operatorname{corr}(\mathbf{w}_1\mathbf{X}_1, \mathbf{w}_2\mathbf{X}_2) \\ &= \arg\max \frac{\mathbf{w}_1\mathbf{\Sigma}_{12}\mathbf{w}_2}{\sqrt{\mathbf{w}_1\mathbf{\Sigma}_{11}\mathbf{w}_1\mathbf{w}_2\mathbf{\Sigma}_{22}\mathbf{w}_2}} \end{aligned}$$

The solution to the optimization problem can be readily obtained through eigen-decompositions. Various extensions of CCA broadens its application areas. In (Correa et al. 2010; Zhao et al. 2016), multiset CCA and hierarchical CCA applied on medical modalities consisting of functional magnetic resonance imaging (fMRI), electroencephalography (EEG) and structural MRI (sMRI) facilitates group inference on brain functions and autism spectrum disorder (ASD) diagnosis. In financial engineering,

CCA can be useful for combining historical closing prices and financial technical variables to better predict stock market (Guo et al. 2014). Among extensive research in learning context-specific word representations for natural language processing (NLP), CCA also helps by effectively merging past and future views of contexts (Dhillon, Foster, and Ungar 2011).

Constrained by its simplistic form, the learning power of vanilla CCA is limited, particularly when the optimal mapping is non-linear. Kernel methods, as will be introduced in Section 2.2, can alleviate the limitation by extending the projection to a reproducing kernel Hilbert space. In (Lai and Fyfe 2000), kernel CCA is proposed to first nonlinearly transform data across modalities to a feature space and then perform linear CCA in that space. More recently, deep learning technique, as will be introduced in Section 2.3, is applied on CCA to further improve the scalability and non-linear mapping (Andrew et al. 2013). By utilizing the advantages of neural networks in flexible predictive modeling, this algorithm is able to learn representations with significantly higher correlation.

## 2.2 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. In general, kernel methods induce an implicit mapping into a reproducing kernel Hilbert space (RKHS), through the construction of a positive definite similarity matrix between samples in the input space, and enable model inference in that 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.

When applied to multimodal feature fusion, kernel methods provide a principled

framework for fusing diverse descriptors into a unified feature space through the technique of Multiple Kernel Learning (MKL) (Scholkopf and Smola 2001). In this section, we first review the basic formulation of single kernels. We then describe in detail the procedure to perform fusion under the MKL framework.

### 2.2.1 Kernel Methods

Given the $d-$dimensional input domain $\mathcal{X}$, the kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ induces a RKHS $\mathcal{H}_k$ with the corresponding inner product $< ., . >_{\mathcal{H}_k}$ and the norm $\|.\|_{\mathcal{H}_k}$. For a set of data-label pairs $\{\mathbf{x}_i, y_i\}_{i=1}^n$, where $y_i$ corresponds to the label of the sample $\mathbf{x}_i \in \mathbb{R}^d$, the problem of inferring a predictive model can be posed as the following empirical risk minimization task (Andrew 2000):

$$f_{opt} = \underset{f \in \mathcal{H}_k}{\arg\min} \frac{1}{n} \sum_i \mathcal{L}(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_{\mathcal{H}_k}, \tag{2.1}$$

where $\mathcal{L}$ denotes a chosen loss function and $\lambda$ is the regularization parameter. For example, in kernel ridge regression $\mathcal{L}$ is chosen to be the $\ell_2$ loss while kernel Support Vector Machine (SVM) uses the hinge loss.

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 both general purpose and domain specific kernels have been proposed in the literature, e.g. radial basis function (RBF) kernels, $\chi^2$ kernel (J. Zhang et al. 2007), string (Lodhi et al. 2002), and graph kernels (Vishwanathan et al. 2010). Furthermore, the classical Representer Theorem allows the representation of any optimal function in $\mathcal{H}_k$ as

$$f_{opt}(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i), \tag{2.2}$$

thereby enabling construction of a dual optimization problem for Equation (2.1) based

only on the kernel matrix and not the samples explicitly. This is commonly referred as the *kernel trick* in the machine learning literature.

### 2.2.2 Fusion with Multiple Kernels

The objective of MKL is to learn a combination of base kernels $k_1, ..., k_M$ and perform empirical risk minimization simultaneously (Bach, Lanckriet, and Jordan 2004; Gönen and Alpaydın 2011; Nilsback and Zisserman 2008; Gehler and Nowozin 2009; Yeh et al. 2012). Conical (Sun et al. 2010) and convex combinations (Rakotomamonjy et al. 2008) are commonly considered and efficient optimizers such as Sequential Minimal Optimization (SMO) (Sun et al. 2010) and Spectral Projected Gradient (SPG) (Jain, Vishwanathan, and Varma 2012) techniques have been developed. In an extensive review of MKL algorithms (Gönen and Alpaydın 2011), Gonen *et al.* showed that the formulation in (Cortes, Mohri, and Rostamizadeh 2009) achieved consistently superior performance on several binary classification tasks. Most recent research in MKL focus on improving the multi-class classification performance (Cortes, Mohri, and Rostamizadeh 2013) and effectively handling training convergence and complexity (Orabona, Jie, and Caputo 2012). In (F. Liu et al. 2014), the authors solved the multiple kernel learning problem directly using its primal formulation, with random Fourier features.

The kernel fusion schemes have been generalized further to create localized multiple kernel learning (LMKL) (Gönen and Alpaydin 2008; Kannao and Guha 2016; Moeller, Swaminathan, and Venkatasubramanian 2016) and non-linear MKL algorithms. In (Moeller, Swaminathan, and Venkatasubramanian 2016), Moeller *et. al.*. have formulated a unified view of LMKL algorithms:

$$k_\beta(\mathbf{x}_i, \mathbf{x}_j) = \sum_m \beta_m(\mathbf{x}_i, \mathbf{x}_j) k_m(\mathbf{x}_i, \mathbf{x}_j), \tag{2.3}$$

where $\beta_m$ is the gating function for kernel function $k_m$. In contrast to global MKL formulations where the weight $\beta_m$ is constant across data, the gating function in Equation (2.3) 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 $\beta_m$ is constructed and how the optimization is carried out. For example, in (Gönen and Alpaydin 2008) $\beta_m$ is chosen to be separable into softmax functions. On the other hand, non-linear 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, (Cortes, Mohri, and Rostamizadeh 2009) considered polynomial combination of base kernels and (Zhuang, Tsang, and Hoi 2011) utilized a two-layer neural network to construct a RBF kernel composition on top of the linear combination.

Although MKL and LMKL provide 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, the optimizations do not produce consistent performance improvements over a simple baseline kernel constructed as the unweighted average of the base kernels (Gehler and Nowozin 2009; Cortes, Mohri, and Rostamizadeh 2009). Furthermore, extending MKL techniques, designed primarily for binary classification, to multi-class classification problems is not straightforward. In contrast to the conventional one-vs-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 (Zien and Ong 2007; Cortes, Mohri, and Rostamizadeh 2013).

## 2.3 Multimodal Deep Learning

As a sophisticated representation learning paradigm, deep learning have shown exceptional power when dealing with complex, high-dimensional data. This is evidenced by the tremendous success of particular deep architectures in several key application areas: Convolutional Neural Network (CNN) (Krizhevsky, Sutskever, and Hinton 2012) and further improved architectures (Szegedy et al. 2015; He et al. 2016) closed the gap with human on ImageNet (Deng et al. 2009) object recognition and detection; Recurrent Neural Networks (RNN), particularly the ones based on Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) have made big strides in speech recognition (Graves, Mohamed, and Hinton 2013), NLP (Mikolov et al. 2010; Sutskever, Vinyals, and Le 2014) and time series analysis (Lipton et al. 2015; Che et al. 2016).

### 2.3.1 Joint Representation Learning for Fusion

The strong representation learning capability of deep learning is also well suited for solving the multi-modal fusion problems. As one of the earliest attempts to perform multimodal deep learning, (Ngiam et al. 2011) constructed a multimodal autoencoder to optimize the combined reconstruction error while (Srivastava and Salakhutdinov 2012) built a multimodal restricted Boltzmann machine to learn a joint density model over the space of heterogeneous inputs. Applied on image-text and audio-video data, such frameworks first construct separate fully connected networks on each modality for individual representation learning and subsequently merge together the hidden representations learned at mid-level or last several layers. A more flexible framework proposed outside of these applications is (Zhao, Hu, and Zhou 2015), where Zhao *et.al.* built sub-networks for each heterogeneous feature and relied on the Stacked Denoising

Autoencoders to learn high-level homogeneous representations for feature integration. Note that the above mentioned approaches learned the joint representations in an unsupervised manner. Although such generic frameworks can be easily extended to a wide range of problems and do not require annotated data, they fail to leverage the end-to-end learning advantage of several specific deep architectures.

Two fusion settings with deep architecture were discussed in (Ngiam et al. 2011; Srivastava and Salakhutdinov 2012): After joint training on two modalities, the network is expected to predict from a single modality while the other one being absent; When both modalities are present, the fused feature aims to better exploit the complementary information contained across modalities. In this dissertation, we mainly focus on the second case where all feature sources are accessible.

### 2.3.2 Joint End-to-End Learning for Fusion

The fusion power of deep learning can be better exploited when used under the end-to-end learning setting. In several important application domains, it has been shown that embeddings learned with deep neural networks can significantly boost or even completely replace conventional hand-engineered features. For example, the visual descriptors learned by CNN are largely preferred over the famous hand-crafted SIFT and HOG features in object recognition and video analysis (Karpathy et al. 2014; He et al. 2016). In audio analysis and speech modeling, RNN and LSTM can learn rich temporal representations which supersede conventional MFCC feature (Graves, Mohamed, and Hinton 2013; Sutskever, Vinyals, and Le 2014). Constructed with carefully designed neural operations and the stacking of a large number of layers, these networks automatically learn multiple levels of abstraction from data which are also specific to the given task. The learning flexibility opens up huge opportunity for fusion tasks.

Existing work in computer vision consider fusing the representations learned from either the same input or different data sources. Focused solely on image classification, Lin *et.al.* utilized two CNNs as the learning model trained on the same set of images (Lin, RoyChowdhury, and Maji 2015). While one network serves as the part detector, the other CNN extracts hierarchical visual descriptors. Consequently, the fusion operation becomes the core to the efficacy in exploiting the correlation between the two sets of hidden representations. While in (Lin, RoyChowdhury, and Maji 2015) a bilinear operation was directly carried out between the two feature matrices at each image location, (Gao et al. 2016; Fukui et al. 2016) demonstrated that such bilinear features are closely related to polynomial kernel and therefore, kernalized analysis can be utilized to significantly simplify the calculation and reduce the fused dimensionality.

When multiple DNNs are trained on different feature sources, the system can gain the capability to handle more complex dependencies in data. For example, in order to handle the spatial-temporal dependencies in videos for action recognition, Simonyan *et.al.* proposed a two-stream network (Simonyan and Zisserman 2014) where one network learns the spatial features from RGB images and the other stream is a temporal network learning on optical flow inputs extracted from the video. In (Park et al. 2016), the authors extended this two-stream framework with specific fusion strategies: a early fusion method to amplify the CNN intermediate representation with the magnitude of the optical flow, and a late fusion mechanism to amplify or suppress the activations of the two networks based on their agreement. The central idea is for the CNN to draw cross-modality information from the temporal network in earlier stages, for example, which part of the image is moving, in order to improve its focusing regions of learning across time. In another application of heterogeneous data consisting of continuous physiological signals and discrete events (Martinez and

Yannakakis 2014), Martinez *et.al.* proposed a pooling fusion scheme which attenuates the modality of densely sampled signals around the events in the discrete modality.

Deep networks are known to be difficult to train. Consequently, several highly effective training strategies have been proposed to avoid model overfitting (Srivastava et al. 2014), reduce covariate shift (Ioffe and Szegedy 2015) and alleviate the learning degradation problem (He et al. 2016). When applying DNNs for fusion, specific training strategies can improve the fusion performance as well. In (Neverova et al. 2016), a special multimodal dropout mechanism is designed to learn cross-modality correlations while prohibiting false co-adaptations in order to improve the model robustness for gesture recognition. Specifically, after joint training over the shared layers across modalities, at each step each modality component is dropped with certain probability.

### 2.3.3 Bridging Deep Learning with Kernel Methods

Despite these recent successes, deep architecture possesses several limitations too. In particular, the increasing complexity of the networks requires exhaustive tuning of several hyper-parameters in the discrete space of network architectures, often resulting in sub-optimal solutions or model overfitting. This is especially common in applications characterized by limited dataset sizes and complex dependencies in the input space. Despite the advances in regularization techniques and data augmentation strategies (Krizhevsky, Sutskever, and Hinton 2012), 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, non-linear representation learning models is to employ kernel machines, which is described in Section 2.2. This motivate a recent wave of research efforts which attempt to incorporate ideas from deep learning

into kernel machine optimization (Cho and Saul 2009; Zhuang, Tsang, and Hoi 2011; Wiering and Schomaker 2014; Wilson et al. 2016; Mairal 2016).

One of the earliest approaches in this direction was developed by Cho *et. al* (Cho and Saul 2009), 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, Tsang, and Hoi 2011). The connection between kernel learning and deep learning can also be drawn through Gaussian processes as demonstrated in (Wilson et al. 2016), where Wilson *et al.* derived deep kernels through the Gaussian process marginal likelihood. Another class of approaches directly incorporated kernel machines into Deep Neural Network (DNN) architectures. For example, Wiering *et al.* (Wiering and Schomaker 2014) constructed a multi-layer SVM by replacing neurons in multi-layer perceptrons (MLP) with SVM units. More recently, in (Mairal 2016), 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 super-resolution showed that the new type of network achieved competitive and sometimes improved performance as compared to CNN.

## 2.4 Multivariate Analysis for Fusion on Sequential Data

Sequential data come in the form as sentences, time series and audio waveforms etc. Capturing the short- and long-term dependencies is central to analyzing these types of data and many of the aforementioned generic techniques are not specifically designed to perform such temporal modeling. Furthermore, real-world sequential signals in natural languages, healthcare and financial domain often contain multiple

time series and the strong correlations between them cannot be neglected. As a result, it is critical to adopt multivariate sequential analysis in order to build highly effective predictive models. Depending on the applications, different multivariate modeling techniques are needed to explore both the temporal and spatial dependencies. In this section, with emphasis on the important applications of clinical time series analysis and speech modeling, we describe several powerful tools for analyzing and fusing the multivariate sequential data.

### 2.4.1 Multivariate State-Space Models

As classic sequence modeling techniques, state-space models have been successfully applied to a wide range of problems in control systems and machine learning (Bishop 2006). The power of state-space models can be attributed to explicitly representing the interactions between system hidden state and actual observations. Generally speaking, a state-space model consists of a set of state equations which describe the evolution of the system hidden state with respect to time and another set of observation equations which describes at each time step how the observations depend on the state. Although most state-space models were originally defined for univariate sequences, the extensions to multivariate cases have been developed over the years (Vlachos and Kugiumtzis 2008; Ryali et al. 2011).

For continuous latent variables, the notable examples of state-space models are Kalman filter under the assumption of linear Gaussian latents. Aiming at solving the filtering and forecasting equations of the state-space, Kalman filter is able to update the state once a new observation is received without reprocess the entire dataset (Sumway and Stoffer 2006). When the states are discrete-valued, one can typically use Hidden Markov Model (HMM) assuming that the specific regimes inside the associated Markov chain are unknown to the observer.

One of the most important applications of state-space models are in clinical modeling. For example, a hierarchical Kalman filter is defined in (Liu and Hauskrecht 2013) to model the linear transition between consecutive states for blood test data. In order to solve the irregular sampling issue existing in clinical data, a secondary Gaussian Process (GP) is further defined over time windows and used to control the higher-level state-space model. In (Ghassemi et al. 2015), a multivariate dynamic system model is derived to estimate the intrinsic causal interactions between distributed brain areas. The state-space formulation allows the system to explore the underlying neuronal activity given only the observed blood oxygen signals in functional MRI. The parameters in both state-space models of (Liu and Hauskrecht 2013; Ghassemi et al. 2015) are calculated using the Expectation-Maximization (EM) algorithm as maximum likelihood estimates. Moreover, targeted specifically at the multivariate nature of clinical measurements, a multi-task GP method is proposed in(Ghassemi et al. 2015) to jointly transforms the measurements into a unified latent space.

Speech signals are another natural application of state-space models when considering, for instance, the audio waveform (or the extracted features on it) as observations whereas the phonemes as the latent states. Many earlier approaches are centered around using HMM for automatic speech recognition (Rabiner 1989; Gales and Young 2008). At its basic form, the HMM models the base phones as the hidden states and the estimated transition probability is used to predict the next acoustic vector. Beyond speech recognition, more sophisticated state-space models have been developed for a wide range of speech modeling tasks including as speech enhancement (Varga and Moore 1990) and speaker diarization (Fox et al. 2011).

### 2.4.2 Deep Learning based Sequence Modeling

As described in Section 2.3, the learning techniques have been revolutionized by the advances of deep architectures in many domains. Sequence modeling is no exception. In recent years, RNN and LSTM architectures have become the *de facto* solution to various temporal modeling tasks. Under the recurrent structure of RNN, output of the network travels along the feedback loop and gets transmitted to the next time step for modeling, thus allowing the sequential information to persist. RNN has been shown to be effective at capturing short-term temporal correlation, but may suffer at long-term dependencies. As comparison, LSTM assigns dedicated hidden layers to selectively decides which part of the information flows through the system state, how the state is updated and what output the system should generate (Hochreiter and Schmidhuber 1997). The superior capability of modeling long-term dependencies has made LSTM the sought-after solution for sequence modeling problems.

Recent efforts utilizing these deep learning based models have demonstrated much improved performance over conventional feature-based models including state-space model and logistic regression. In the clinical modeling area, the earliest work in this direction was by Lipton *et. al.* (Lipton et al. 2015), which proposed to use LSTMs with additional training strategies for diagnosis tasks. In (Lipton, Kale, and Wetzel 2016), RNNs are demonstrated to automatically deal with missing values when they are simply marked by an indicator. In order to learn representations that preserve spatial, spectral and temporal patterns, recurrent convolutional networks have been used to model EEG data in (Bashivan et al. 2015). After the introduction of the MIMIC-III datasets, (Harutyunyan et al. 2017) have rigorously benchmarked RNNs on all four clinical prediction tasks and further improved the RNN modeling through joint training on all tasks.

In speech modeling tasks, LSTM networks with deeper layers (Graves, Mohamed, and Hinton 2013; Sak, Senior, and Beaufays 2014) and special connections (Y. Zhang et al. 2016) have made breakthrough in acoustic modeling and speech recognition problems. Furthermore, by incorporating the strong temporal modeling capability into new architectures in deep metric learning, one can build potentially more superior models for capturing speaker identity information. This has huge implication for several important speech modeling problems including speaker identification, speaker verification and speaker diarization. For example, recently supervised metric learning architectures, namely, *siamese* (Chopra, Hadsell, and LeCun 2005; Koch, Zemel, and Salakhutdinov 2015) and *triplet* (Hoffer and Ailon 2015; Schroff, Kalenichenko, and Philbin 2015) networks have been developed to automatically infer similarity metrics to compare speech segments. Broadly speaking, these architectures infer a non-linear mapping $\mathcal{A}(\cdot)$, such that, in the resulting latent space the within-class sample distances are minimized while the between-class distances are maximized based on a certain margin. In (Le Lan et al. 2017), Lan *et al.* proposed to employ triplet networks on i-vectors to infer a similarity metric, and achieved state-of-the-art results over conventional metrics in the diarization literature. Despite its effectiveness, it is important to note that the feature extraction process is disentangled from the metric learning network and hence cannot support end-to-end inferencing. In (Garcia-Romero et al. 2017), a joint learning is developed from the MFCC features directly under the siamese network setting. However, compare to the triplet ranking loss, which requires the margin to be satisfied only a each given inference sample, the siamese loss requires a global margin for all negative samples and hence exhibits much less flexibility.

Among many recurrent realizations in the NLP domain, another important component which is often employed along with LSTM is attention mechanism. As an

integral part to achieve the superior performance, attention is often placed between LSTM encoder and decoder in sequence to sequence architectures (Bahdanau, Cho, and Bengio 2014; Xu et al. 2015; Vinyals et al. 2015; Hermann et al. 2015). However, recent research in language sequence generation indicates that by stacking the blocks of solely attention computations, one can achieve similar performance as RNN and LSTM (Vaswani et al. 2017). Besides providing significantly faster training, attention networks demonstrate efficient modeling of long-term dependencies. These recent advances illustrate the promising value of attention computations in sequential modeling.

## 2.5 Multilayer Graphs for Fusion on Relational Data

In previous sections, we focus on building the fusion model from the data samples directly. In many real-world applications, however, there usually exist important relational information between the samples. Furthermore, in some cases, deriving the features for each sample can be difficult whereas access to the relations is readily available. Graph is a natural data structure to represent and analyze such relational data and hence the modeling, fusion and inferecing on graph-structured data have become central to a wide-range of machine learning problems.

The most common tool for handling graph structured data often represent data as undirected graphs with vertices representing entities and edges describing the relationships between the entities. For example, each account in a social network can be represented as vertices whereas the connections and interactions are modeled as edges. The edges can additionally be associated with weights indicating the strength of the affinity, e.g. the frequency of conversation between the persons or the number of topics they are commonly interested in. Another example is representing the functional or structural regions of the brain as vertices of a graph and the

relationships between the various regions as the edges. Modeling such relationships is a critical step towards understanding, diagnosing and eventually treating a gamut of neurological conditions including epilepsy, stroke, and autism (Kelly et al. 2008). Note that while the edge weights are often considered as the important attributes, values associated with vertices, such as the time-varying signals detected in each brain region, can also enable encoding of additional properties about the entities (Shuman et al. 2013). These inherent modeling flexibility, the significance of the applications and the development of large-scale intelligent algorithms have made graph-based inferencing very attractive (Cook and Holder 2006).

When the graph structure exists in multiple data sources or the extracted features, it is beneficial to study the relationships as a whole in order to obtain an holistic understanding. In such scenarios, the relationships can be easily represented by multiple undirected graphs with the same set of vertices and edges arising from different attributes. Depending on the research area, this type of data structure is referred to as multilayer graph (Dong et al. 2012; Kivelä et al. 2014), multidimensional network (Boutemine and Bouguessa 2017; Berlingerio et al. 2011) or multiplex network (Mucha et al. 2010). To be consistent, we adopt the notation of *multilayer graphs* in this dissertation, with each layer corresponding to one relational representation arising from the attributes. The heterogeneity in the relationships, while providing richer information, makes statistical inferencing challenging. Furthermore, the varying levels of sparsity in different layers and the inherent uncertainties in neighborhood graphs, e.g. noisy edges or outliers, add to the complexity of this problem. Consequently, effective fusion mechanism of the heterogeneous relationships is necessary to construct a high-fidelity inferencing model.

### 2.5.1 Applications of Multilayer Graph Fusion

Multilayer graph fusion is very useful in a large number of machine learning topics including node classification, link prediction and community detection. Node classification is one of the most common applications. In this problem, each node is assigned one class label (Bhagat, Cormode, and Muthukrishnan 2011; Kipf and Welling 2016) or multiple labels (i.e. multi-label classification) (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016). The model can be learned either from the nodes on the same graph as the testing nodes, or from nodes on entirely different graphs. These specific problems are named transductive learning and inductive learning respectively (Joachims 2003; Yang, Cohen, and Salakhutdinov 2016; Hamilton, Ying, and Leskovec 2017; Velicković et al. 2017). In general, node classification algorithms first extract the graph embedding as vector representations for each node and then learn a classifier on them. Some recent work unifies these two steps to learn the graph node representations which are specific to the classification problem at hand (Kipf and Welling 2016; Monti et al. 2017).

In contrary to node classification which perform inferencing on graph nodes, in link prediction, the model is supposed to infer the graph edge correctly. This application is useful particularly for social networks where some important connections between friends have not been established. Interestingly, the graph embedding which are usually learned for node classification have been proven helpful also for link prediction (Liben-Nowell and Kleinberg 2007; W. Liu et al. 2017). The underlying reason is that the learned representations capture rich graph structural information and different orders of proximity such as the missing link can be effectively inferred.

Community detection is the problem of partitioning the graph vertices into several groups such that the connections within each community is dense whereas the connec-

tions across communities are sparse. As an unsupervised learning technique, it is an extremely useful analysis tool for real-world large-scale networks where the ground-truth node labeling is absent. In order to obtain desirable community structures, it has been shown that the mesoscopic information is equally important as the lower-order proximity captured by common graph embedding techniques (X. Wang et al. 2017; Cavallari et al. 2017). In the case of multilayer graphs, each layer characterizes a specific kind of relationship and they can be aggregated to better investigate the community structure. For example, in the AUCS dataset (Magnani, Micenkova, and Rossi 2013) and MIT reality mining dataset (Eagle and Pentland 2006), the nodes correspond to persons and different layers represent their interactions at work, through phone call connections and on social networks. By jointly exploiting these relational information, we can obtain communities such as their department affiliation or study interest groups. A comprehensive survey studying the community detection algorithms and the multilayer graph datasets for this research can be found in (Kim and Lee 2015). In the following sections, we focus on the community detection problem and describe in detail several important algorithms to tackle it.

### 2.5.2 Modularity-based Multilayer Community Detection

Many community detection algorithms depend on heuristic measures to define good partitioning. A prominent measure used in the context of multilayer graphs is multislice modularity (Mucha et al. 2010). Originally defined for single graphs, modularity measures the difference between the number of edges running across groups and the one as expected from random edge assignment. Formally modularity is defined as (Newman 2006):

$$Q = \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(g_i, g_j) \tag{2.4}$$

where $A_{ij}$ denotes the connection between node $i$ and $j$, $k_i, k_j$ denotes the degree for node $i$ and $j$ respectively, $m = \frac{1}{2}\sum_i k_i$ is the total number of edges and $\delta(\cdot)$ is the Kronecker delta function which equals one only when the community membership for node $i$ and $j$, namely $g_i$ and $g_j$, is the same. It is clear that only the pairwise relations of nodes belonging to the same community will contribute to modularity. This contribution will be positive when the edge weight $A_{ij}$ is larger than the expected edge weight $\frac{k_i k_j}{2m}$. Conversely, it will be negative when $A_{ij} < \frac{k_i k_j}{2m}$.

Some popular algorithms for solving the maximization of $Q$ include spectral decomposition (Newman 2006), greedy methods (Blondel et al. 2008), extremal optimization (Duch and Arenas 2005) and mathematical programming (Agarwal and Kempe 2008). For an extensive survey on existing optimization algorithms, please refer to (Chen, Kuzmin, and Szymanski 2014). To extend these successful detection algorithms to multigraphs, (Mucha et al. 2010) derived the multislice modularity based on Laplacian dynamics. As a result, the conventional modularity maximization algorithms can be conveniently adopted to this new measure to solve the multilayer partitioning problem. According (Mucha et al. 2010), the multislice modularity can be defined as:

$$Q_{\text{multi}} = \frac{1}{2\mu}\sum_{i,j}\sum_{d,r}[(A^d_{ij} - \gamma_d \frac{k^d_i k^d_j}{2m_d})\delta(d,r) + \delta(i,j)\sigma^{d,r}_j]\delta(g^d_i, g^r_j) \tag{2.5}$$

where $\mu$ is the normalization factor, $\gamma_d$ is the resolution parameter for layer $d$, $\sigma^{d,r}_j$ is the coupling parameter of node $j$ between layer $d$ and $r$, and other parameters are simply the extensions from Equation (2.4) to layer $d$.

As observed in Equation (2.5), $Q_{\text{multi}}$ separates the contribution of intralayer connections (first term) and interlayer coupling (second term). The intralayer modularity contribution is similar to Equation (2.4) whereas the interlayer contribution measures the connectivity strength from node $i$ to all its corresponding nodes in other layers. As shown in (Mucha et al. 2010), by increasing the coupling parameter $\sigma$, one obtains fewer communities from the multilayer graph and at the same time, the average number of communities for each node reduces. Besides being successful optimization objective measures, modularity and multi-slice modularity are widely utilized as evaluation metrics for benchmarking community detection algorithms in the literature (Almeida et al. 2011; Tagarelli, Amelio, and Gullo 2017; Boutemine and Bouguessa 2017)

### 2.5.3 Graph Aggregation and Ensemble Clustering

Different from above algorithms which develop special strategies to tackle the multiple relations from scratch, graph aggregation and ensemble clustering techniques aim at deriving the fusion solution based on single graph clustering results. The fusion procedure can be applied at different stages of the algorithm pipeline. In graph aggregation, different layers are combined based on some metric into a single graph which encompasses the holistic view of all types of relations. On the contrary, ensemble clustering works directly on the pre-defined partitions of each layer and aggregate them to construct a clustering consensus.

One of the earliest work on graph aggregation is (Berlingerio, Coscia, and Giannotti 2011), which develops heuristic rules to obtain fused edge weights. In the case of binary aggregation, there will be an edge in the aggregated graph if the edge exists in at least one layer. Another flattening rule is frequency-based: the resulting edge weight is the number of edges connecting the two nodes in different layers, normalized

by a constant factor. Although these rule-based aggregation schemes are efficient and easy to implement, they suffer from some serious drawbacks: both treat the weighting of each layer equally and it is not clear which strategy should be adopted given a new set of data. To tackle these issues, recent work focus on developing more sophisticated flattening rules based on local relations. For example, in (Kun, Caceres, and Carter 2014), the aggregation rule is inspired by boosting: the edge weights in each layer are treated as weak similarities and a reward system promoting presence of strong associations and absence of week edges is developed. In (Kim, Lee, and Lim 2017), the contribution of each layer is automatically determined by maximizing the clustering coefficient computed from the combined graph. The resulting differential flattening optimization is solved by interior point methods.

Ensemble clustering positions the fusion procedure in a late stage. Algorithms of this type usually work directly on an ensemble of existing partitioning from every layer and thus do not make assumption of the inherent clustering algorithm applied on each graph. For example, based on the community structures of each layer, (Berlingerio, Pinelli, and Calabrese 2013) builds a transaction list where each node is labeled by a pair consisting of its dimension (layer) and the community it belongs on that dimension. By drawing the analogy of the community membership as items and the nodes as the transactions of these items, it applies the frequent closed itemset mining algorithm to find communities with large overlap across different layers. Along the same line of thinking is (Tagarelli, Amelio, and Gullo 2017), where an ensemble clustering is derived based on the pre-defined partioning of each graph layer. Different from conventional ensemble clustering algorithms which often solely rely on the co-occurrence of community assignments across feature sources, in the case of multilayer graph, the original relations between samples are known and can be exploited to

achieve the optimal consensus. To this end, the authors proposed to derive the consensus such that the topology of the multilayer graph is preserved by maximizing the multilayer modularity. Starting from the consensus community structure provides by conventional ensemble clustering, at each iteration, the algorithm refines the within-community and across-community connectivity, possibly on different layers, in order to increase the modularity score. It should be noted that although both ensemble approaches mentioned here aim to fuse community structures from different layers, their problem statements are different: (Tagarelli, Amelio, and Gullo 2017) gives a single final partition of the graph nodes, whereas (Berlingerio, Pinelli, and Calabrese 2013) can discover different multilayer communities, i.e. each multilayer community may reside on a subset of the layers and each node may belong to different multilayer communities depending on the layer it is considered.

### 2.5.4 Representation Learning based Approaches

The objective of representation learning based approaches is to derive vector representations for each node which capture the proximity and structural information, such that conventional clustering algorithms e.g. $k$-means can be directly applied to obtain the partition. The success of representation learning in other significant domains ranging from natural language processing to computer vision has made such topic very attractive for relational data. A popular example following this procedure is spectral clustering, which infers a subspace representation from the eigen-spectrum of the graph Laplacian and subsequently applies $k$-means clustering (Ng, Jordan, and Weiss 2001). More recently, there has been particular interest in adopting neural word embeddings, which encode semantic relations in a continuous vector space with a relatively small number of dimensions, to the case of graphs. At its core, word embedding algorithms build on the distributional hypothesis (Harris

1954) where similar contexts imply similar meanings, i.e., co-occurrences of words can be tied to their underlying meanings. Extending this principle to the case of arbitrary networks requires the definition of the notion of co-occurrence between vertices. Recent approaches such as *DeepWalk* (Perozzi, Al-Rfou, and Skiena 2014) and *Node2Vec* (Grover and Leskovec 2016) address this challenge by creating a stream of randomly generated walks between the vertices and generalize word embeddings to graphs. The resulting latent representations are highly robust, can capture the structural regularities and recover the community memberships.

In order to exploit and fuse the multiple relations, specific changes are needed to extend above single graph representation learning to multilayer graphs. Several existing approaches tackle this problem using factorization on the graph adjacency matrices (Tang, Lu, and Dhillon 2009; Dong et al. 2012; Gligorijević, Panagakis, and Zafeiriou 2016; Papalexakis, Akoglu, and Ience 2013). In (Tang, Lu, and Dhillon 2009; Dong et al. 2012), either the adjacency matrix $\mathbf{A}^{(m)}$ or the Laplacian matrix $\mathbf{L}^{(m)}$ corresponding to layer $G^{(m)}$ is decomposed into a set of shared eigenvectors among all layers and layer-specific eigenvalues. In (Gligorijević, Panagakis, and Zafeiriou 2016), the symmetric non-negative matrix tri-factorization algorithm is utilized and each $\mathbf{A}^{(m)}$ is factorized into 3 non-negative matrices including a shared cluster indicator matrix. Its formulation for multilayer graph also deals with incomplete and missing layers by only factorizing observed entries.

Utilizing the word-embedding based approaches to combine information from multiple layers has not been explored yet in the literature. Based on DeepWalk or Nede2Vec, one can obtain a straightforward extension to the case of multi-layer graph for feature fusion: (a) Perform random walks on each of the layers independently and infer the latent representations using deep neural embeddings, (b) Merge the latent

features, e.g. concatenation, from all layers and use the resulting unified representation for clustering tasks. However, by performing independent random walks, this approach does not exploit the common factors or regularity in the community structures across the different layers. Furthermore, the commonly adopted late fusion strategies such as concatenation do not compensate for the uncertainties introduced by each of the layers while identifying clusters.

# Chapter 3

## MULTIMODAL SENSOR FUSION AND ENSEMBLE INFERENCE

The use of mobile devices and wearable sensors for activity monitoring has become an important research problem in the recent years. The objective of the sensor based intelligence system is to analyze the data collected using the inherent sensing modalities and obtain predictive inferences using low complexity algorithms. The challenges in building effective predictive algorithms for such mobile devices are twofold. On one hand, though abundantly available, the data collected from these cheap sensors are often very noisy and unreliable. On the other hand, it can be prohibitive in terms of both time and resource availability, to employ complex machine learning techniques to process this data. While there have been significant advances in activity recognition using data from high-performance, stand-alone sensors attached to human body (Avci et al. 2010; Zhang and A. A Sawchuk 2013), adapting them to the case of low-cost, mobile sensors is not straightforward.

In spite of the availability of different sensors, activity recognition is often carried out solely based on accelerometer data (Kwapisz, Weiss, and Moore 2011). Using data from other sensors can potentially learn more effective representations, and make the predictor highly robust to measurement inaccuracies. In this chapter, we propose a two-stage systematic fusion framework in order to improve smartphone sensor based activity recognition. One important feature of the proposed approach is the support for both multiple sensing modalities and multiple feature extraction strategies.

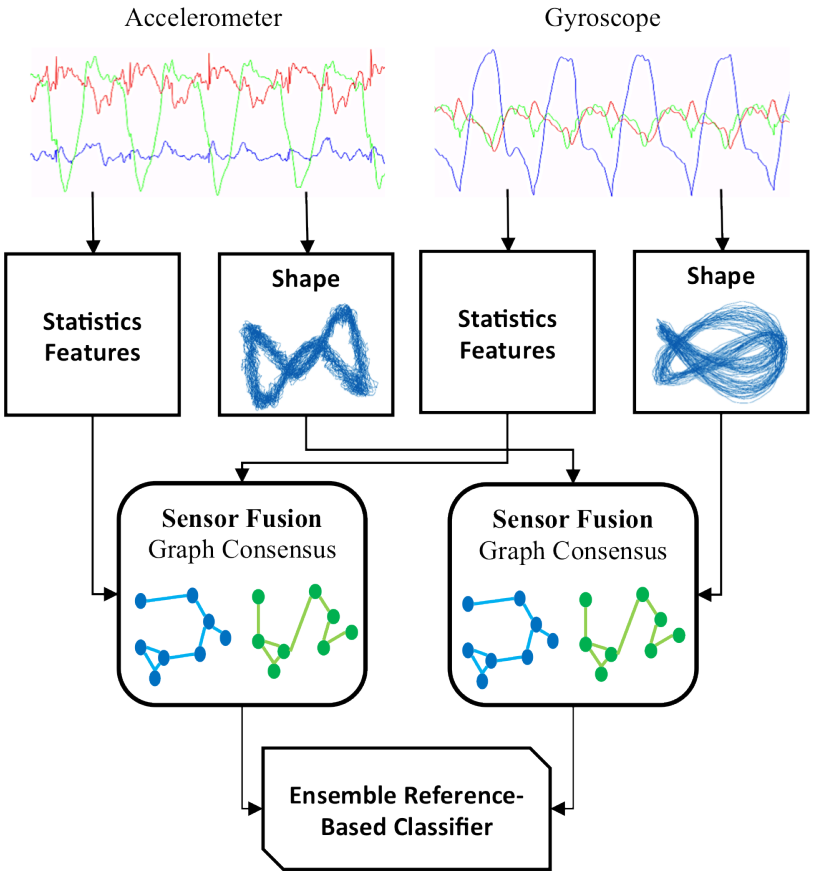



Figure 1: Proposed two-stage architecture for activity recognition on mobile devices.

### 3.1 Multi-modal Consensus Framework

As shown in Figure 1, the proposed framework consists of three steps: (a) extract statistical and shape features from segments using windowing, (b) perform sensor fusion for each feature type across all modalities based on multilayer graphs, and (c) use ensemble reference-based classifier on the different types of fused feature for recognition. The last two steps constitute the fusion stages.

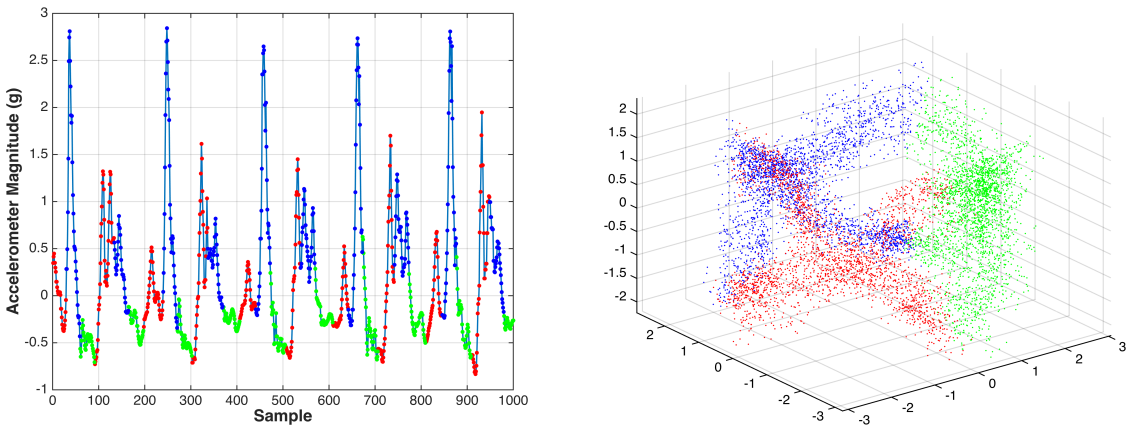
#### 3.1.1 Feature Extraction

We extract statistical features that have proved to be useful for activity recognition (Zhang and A. A Sawchuk 2013; Kwapisz, Weiss, and Moore 2011; Zhang and A. A

Sawchuk 2011), and investigate the time delay embedding of the activity signals and propose to use a basic version of shape features.

The statistical features we extracted are: mean, median, standard deviation, kurtosis, skewness, total acceleration, mean-crossing rate, autoregressive (AR) coefficients (Spanias 2014) and dominant frequency. Each activity signal was first windowed into 5 second non-overlapping segments. This length was chosen empirically such that there is sufficient periodic structure in each segment. The AR coefficients were extracted assuming each segment to be a stationary random signal (He and Jin 2008). The model order was determined to be 3 based on the Akaike information criterion (Akaike 1974). The dominant frequency is defined as the frequency component having the largest FFT magnitude (Zhang and A. A Sawchuk 2011). These statistical features were extracted separately from signals corresponding to the three axes of each sensor and then concatenated together. For both the accelerometer and gyroscope, the overall dimension of the statistical features is 31.

Given a short sequence of measurements, time-delay embedding (TDE) (Sauer, Yorke, and Casdagli 1991) is an approach for reconstructing the underlying system dynamics. Two important parameters for calculating the TDE are: the dimension of reconstruction space $m$ and time delay $\tau$. Given a time series $\mathbf{o}$, the TDE can be represented as a matrix $\mathbf{O}$ whose $i$th column is $[o_i, o_{i+\tau}, o_{i+2\tau}, ..., o_{i+(m-1)\tau}]$. Figure 2 visualizes the raw accelerometer signal for fast walking and its corresponding TDE representation. In Figure 2(b), we cluster the samples in the 3-D PCA representation of TDE and mark different clusters with specific colors. The corresponding activity samples are then marked the same color and illustrated in Figure 2(a). Notice that across the periods of the activity signal, the clusters map to very similar regions. This




(a) (b)

Figure 2: Extracting shape features - (a) Raw accelerometer data, (b) 3-D PCA representation of its delay embedding.

shows that TDE represents the periodic structure of the signal as desired and we can derive suitable features from it for the classification task.

We extract a simple shape function based on the geometric distance property, and use it to derive our feature. The shape function we consider measures the pairwise distance between samples in the TDE space, calculated as $\mathbf{S}_{ij} = \|\mathbf{o}_i - \mathbf{o}_j\|_2$ (Venkataraman et al. 2013). A histogram is constructed on these distances with specified bin size to obtain the feature.

### 3.1.2 Sensor Fusion using Multilayer Graph

The goal of the sensor fusion is to obtain a unified feature for each activity segment by fusing similar features from the two modalities (accelerometer and gyroscope). We adapt the multilayer graph consensus approach in (K.N. Ramamurthy et al. 2014), where each layer represents a single modality containing an intra- and inter-class graph corresponding to the class relationships of the activities. We estimate linear local discriminant embeddings (LDE) instead of kernel embeddings on the graphs to

keep the process computationally simple. Figure 3 shows the overview of this process for a given feature type. Note that we will obtain separate consensus projections for the two feature types, namely statistical and shape.

Denote the $T$ modalities in one feature type as $\{\mathbf{X}_t\}_{t=1}^{T}$, where the columns of $\mathbf{X}_t \in \mathbb{R}^{M_t \times N}$ correspond to the features extracted from each activity segment. The label for an activity segment $i$ is denoted by $l_i$. We construct the intra- and inter-class graphs for modality $t$ and represent the adjacency matrices as $\mathbf{W}_t$ and $\mathbf{W}'_t$, whose elements are defined using the Gaussian RBF with parameter $\gamma$,

$$w_{t,ij} = \begin{cases} e^{-\gamma \|\mathbf{x}_{t,i} - \mathbf{x}_{t,j}\|^2}, & \text{if } l_i = l_j, \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

$$w'_{t,ij} = \begin{cases} e^{-\gamma \|\mathbf{x}_{t,i} - \mathbf{x}_{t,j}\|^2}, & \text{if } l_i \neq l_j, \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

The idea of linear LDE (Chen, Chang, and Liu 2005) is to construct the low-dimensional embedding $\mathbf{V}_t = \mathbf{U}_t^T \mathbf{X}_t$, $\mathbf{U}_t \in \mathbb{R}^{M_t \times D}$ ($D$ is the dimension of projection) being the projection matrix, such that the neighboring points of same class in the ambient space are still close, whereas the neighboring points from different classes are distant. Defining the Laplacian matrices $\mathbf{L}_t = \mathbf{D}_t - \mathbf{W}_t$, and $\mathbf{L}'_t = \mathbf{D}'_t - \mathbf{W}'_t$, where $\mathbf{D}_t$ and $\mathbf{D}'_t$ are the respective diagonal degree matrices, the individual discriminant projections can be computed as the trace-ratio maximization (Jia, Nie, and Zhang 2009),

$$\mathbf{U}_t = \underset{\mathbf{U}_t : \mathbf{U}_t^T \mathbf{U}_t = \mathbf{I}}{\arg\max} \frac{\mathrm{Tr}(\mathbf{U}_t^T \mathbf{X}_t \mathbf{L}'_t \mathbf{X}_t^T \mathbf{U}_t)}{\mathrm{Tr}(\mathbf{U}_t^T \mathbf{X}_t \mathbf{L}_t \mathbf{X}_t^T \mathbf{U}_t)}. \tag{3.3}$$

Since the individual projections belong to the Grassmann manifold, the consensus projection $\mathbf{U}$ can be obtained as geometric mean of the individual projections with

respect to the chordal distance (Ye and Lim 2014),

$$d^2_{proj} = D - \sum_{t=1}^{T} \mathrm{Tr}(\mathbf{U}\mathbf{U}^T\mathbf{U}_t\mathbf{U}_t^T). \tag{3.4}$$

We also require the consensus projection to be discriminative across all the modalities. Combining this with (3.4), the final optimization is

$$\begin{aligned} \min_{\mathbf{U}} \quad & \sum_{t=1}^{T} \mathrm{Tr}(\mathbf{U}^T\mathbf{X}_t\mathbf{L}_t\mathbf{X}_t^T\mathbf{U}) - \alpha \sum_{t=1}^{T} \mathrm{Tr}(\mathbf{U}\mathbf{U}^T\mathbf{U}_t\mathbf{U}_t^T) \\ \text{s.t.} \quad & \sum_{t=1}^{T} \mathrm{Tr}(\mathbf{U}^T\mathbf{X}_t\mathbf{L}'_t\mathbf{X}_t^T\mathbf{U}) = c, \mathbf{U}^T\mathbf{U} = \mathbf{I} \end{aligned}$$

where $\alpha$ is the trade-off parameter. This can be posed as the trace-ratio maximization,

$$\max_{\mathbf{U}:\mathbf{U}^T\mathbf{U}=\mathbf{I}} \frac{\mathrm{Tr}\left(\mathbf{U}^T\left(\sum_{t=1}^{T}\mathbf{X}_t\mathbf{L}'_t\mathbf{X}_t^T\right)\mathbf{U}\right)}{\mathrm{Tr}\left(\mathbf{U}^T\left(\sum_{t=1}^{T}\mathbf{X}_t\left(\mathbf{L}_t - \alpha\mathbf{U}_t\mathbf{U}_t^T\right)\mathbf{X}_t^T\right)\mathbf{U}\right)}$$

and solved using the decomposed Newton's or the iterative trace ratio method (Jia, Nie, and Zhang 2009). The out-of-sample projection for the test data $\{\mathbf{Y}_t\}_{t=1}^T$ is obtained as $\mathbf{Z} = \sum_{t=1}^{T}\mathbf{U}^T\mathbf{Y}_t$.

### 3.1.3 Ensemble Reference-Based Classification

Given different types of consensus features, it is important that the classification mechanism can effectively combine them. We extend the reference-based classification in (Q. Li et al. 2013) using an ensemble classification approach. Different from (Q. Li et al. 2013), we use the whole training data as the reference set and we perform inference directly based on the combined similarity matrix between a probe sample and the reference set. This simplifies the classification and also takes into consideration characteristics of both features. The detailed steps are:

1. Denote a probe sample as $\mathbf{z}$ and the $F$ consensus features as $\{\mathbf{V}_f\}_{f=1}^F$. We construct the similarity vector $\mathbf{s}_f$, where each element $s$ is the similarity between

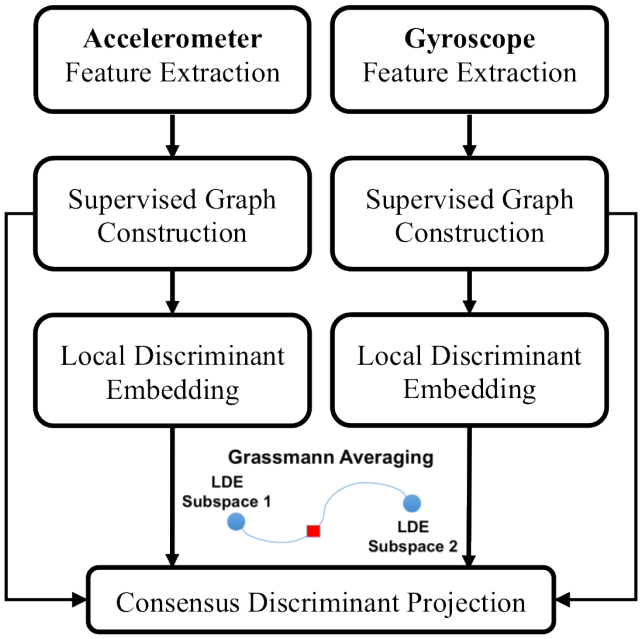



Figure 3: Multilayer graph consensus algorithm for fusing features from two different sensors.

the probe sample and one sample $\mathbf{v}$ in the training feature $\mathbf{V}_f$(Q. Li et al. 2013), $s = 1 - \frac{\gamma(\frac{k}{2}, \frac{d_{\mathbf{z}}^{\mathbf{v}}}{2})}{\Gamma(\frac{k}{2})}$. Here $d_{\mathbf{z}}^{\mathbf{v}}$ is the Euclidean distance between the probe sample $\mathbf{z}$ and the reference sample $\mathbf{v}$. $\Gamma$ is the Gamma function and $\gamma$ is the lower incomplete Gamma function with parameter $k$.

2. Select the top $K = 30$ closest samples from every class and form the new similarity vectors $\{\mathbf{s}_f^{'}\}_{f=1}^{F}$.

3. Denote the elements containing similarities to class $c$ as $(\mathbf{s}_f^{'})^c$. We perform the ensemble for measuring the closeness of the probe sample to class $c$ as, $S^c = \sum_f \sum_n (\mathbf{s}_f^{'})^c$. The inference can then be directly carried out by assigning the label of the class having largest $S^c$ value.

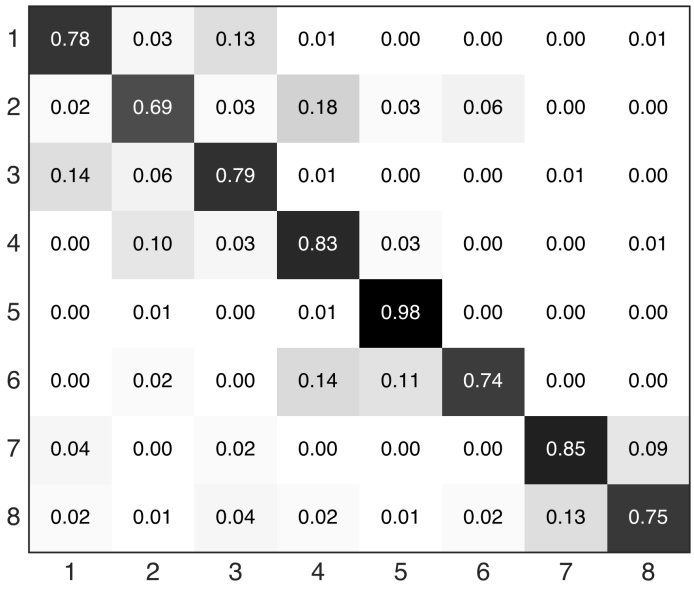



Figure 4: Average confusion matrix of the proposed recongition algorithm.

### 3.2 Experimental Results

Human movement in daily activities are complex in nature. Even for the same activity, the styles can be largely different among people. Hence, we collected data from a set of 32 subjects with diversity in gender, age, weight, and height. The statistics of these demographic factors are shown in Table 1. Each subject performed 5 different activities, - slow walking, fast walking, running, slow biking, and fast biking - using a treadmill or biking machine. The first three activities were performed twice with the subjects carrying the mobile phones first in their front pockets and then in their back pockets, whereas biking activities were performed with mobile phone only in front pockets. As a result, the dataset contains data from 8 classes in total. The labeling also follows this order. The duration of each activity was 75 seconds and the speeds were fixed. The Nexus 4 Android phone that we used had one 3-axis accelerometer, and one 3-axis gyroscope to measure the amount of rotation. We set the sampling rates of the sensors at 200Hz through the Android APK interface.

Table 1: Demographic statistics of the subjects that participated in our data collection experiment.

| **Statistics** | **Mean** | **STD** | **Range** |
|---|---|---|---|
| Age | 30.5 | 7.8 | 20-52 |
| Height (cm) | 174.8 | 9.5 | 155-191 |
| Weight (kg) | 73.9 | 14.0 | 42-108 |

Table 2: Activity recognition performance obtained using different combinations of sensors and features, in comparison to the proposed two stage architecture.

| **Sensor** | **Feature** | **Recognition Rate** |
|---|---|---|
| Accelerometer | Shape (LDE) | 57.8 |
| Gyroscope | Shape (LDE) | 51.66 |
| Shape (Consensus) | | 68.73 |
| Accelerometer | Stats (LDE) | 70.05 |
| Gyroscope | Stats (LDE) | 69.95 |
| Stats (Consensus) | | 73.2 |
| Two Stage Approach | | **80.14** |

We performed 5-fold cross-validation on this dataset by a random split of data according to subject label. In other words, in each validation the training and testing data do not come from the same subject. This setting increases the challenge of the task but better simulates real-world applications.

Table 2 compares the recognition rates in each step of our framework, i.e., using the two sensors independently with each of the features, consensus of the two sensors for each of the features, and finally our two stage architecture. We observed that, with both the feature extraction strategies, sensor fusion performs better than using any sensor alone. However, using the ensemble classifier improves the performance significantly, providing an improvement of around 10% over the best results obtained with a single sensor. Figure 4 plots the confusion matrix for the 8 classes, obtained using the proposed algorithm.

Chapter 4

# MULTIPLE KERNEL FUSION FOR VISUAL CONTEXT MODELING

In visual recognition, it is a common strategy to fuse multiple descriptors which contain complementary information. However, most existing feature fusion algorithms only employed low-level features, i.e., they describe the local variabilities without taking the global context into account. It is known that high-level information, referred to as the context, is crucial to object/scene understanding (Thiagarajan et al. 2014). In general, building a context model is challenging due to both the computational complexity in solving the MAP (Maximum A Posteriori) formulation and the difficulty in modeling complex patterns using limited training data. Consequently, *auto-context* models (Tu 2008) have been developed, which can approximate the posterior using an iterative, supervisory approach. These models integrate the low-level features with context information in the form of probability maps, obtained using a series of classifiers. By enabling the classifier to choose different supporting neighbors to modify the current probabilities towards the ground truth, auto-context methods lead to better regularization.

In this chapter, we propose to adopt auto-context models under the RKHS setting. In addition to providing the flexibility of auto-context models, the proposed approach can build highly effective kernel models for object recognition. Since auto-context probability maps cannot be directly incorporated into the RKHS, we first estimate marginal probabilities using a classifier (e.g. Kernel Logistic Regression or Kernel SVM) and construct an *auto-context* kernel based on these probabilities. For example, the marginalized kernel construction in (Tsuda, Kin, and Asai 2002) can be used.

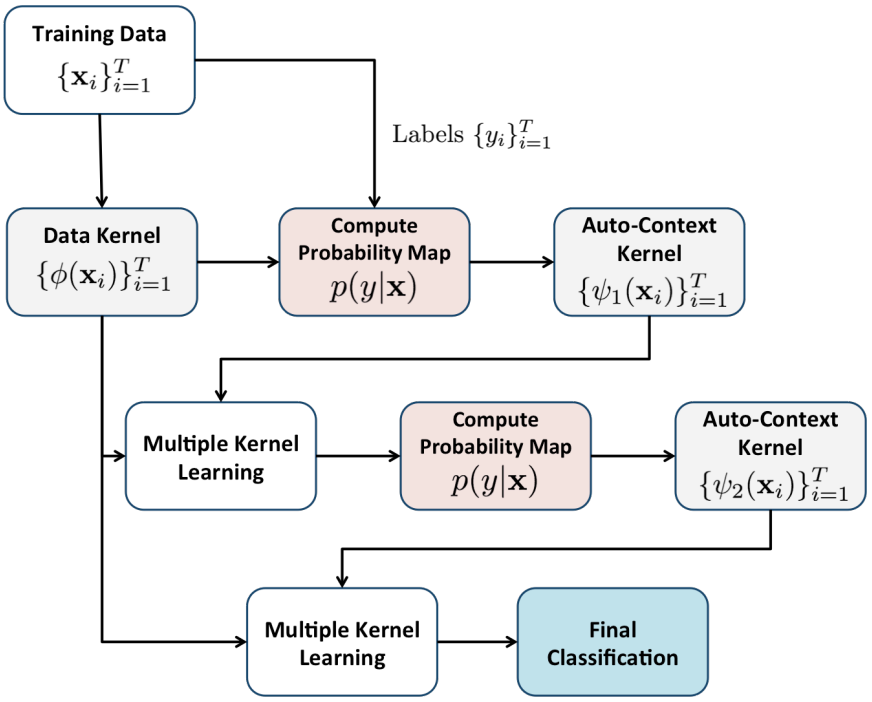



Figure 5: Proposed approach for integrating auto-context with image features under the RKHS setting (illustrated for two iterations). This problem is solved efficiently by posing the fusion in each step as Multiple Kernel Learning (MKL).

Since any symmetric positive definite kernel defines a unique RKHS, we can use other forms of kernels by treating the probability map for each image as a feature vector directly. Interestingly, the process of fusing the auto-context model with the image appearance can be viewed as multiple kernel learning, for which a variety of efficient solutions exist. Figure 5 illustrates the proposed approach with two iterations. We demonstrate using standard object/scene classification datasets that the proposed approach results in highly effective recognition systems.

### 4.1 Formulation of Kernel Methods

Given the feature domain $\mathcal{X} \subset \mathbb{R}^d$, we define the matrix of $n$ samples as $\mathbf{X} = [\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T]$. A function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ defines a valid kernel if it gives rise to a

positive definite kernel matrix $\mathbf{K}$ satisfying Mercer's condition (Schölkopf and Smola 2002). In this case, $k$ also defines an implicit mapping $\varphi$ to the RKHS $\mathcal{H}_k$ and an inner product $\langle \cdot, \cdot \rangle$ in $\mathcal{H}_k$, such that $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_{\mathcal{H}_k}$.

When data from two classes are not linearly separable, it is often beneficial to transform them through the non-linear mapping $\varphi$ to a higher-dimensional space $\mathcal{H}_k$, such that a non-linear decision boundary can be effectively learned using linear classifiers. For example, the RBF kernel maps data into an infinite dimensional RKHS and admits a large class of decision functions, referred as the *native space* (Thiagarajan, Bremer, and Ramamurthy 2014). A crucial advantage of kernel methods is that they do not require an explicit definition of the mapping $\varphi$ and utilize the dual formulation of optimization problem defined solely based on the kernel matrix $\mathbf{K}$ (Andrew 2000). For example, the kernel SVM formulation can be expressed as

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t. } 0 \leq \alpha_i \leq C, \forall i; \sum_i \alpha_i y_i = 0. \end{aligned} \tag{4.1}$$

where $\alpha_i$ are the Lagrangian multipliers, $C$ is the misclassification trade-off parameter and the kernel $k$ is pre-defined by the user.

Since choosing the right kernel for an application is not straightforward, it is common to consider multiple kernels based on different kernel similarity constructions or feature sources for the data. In such scenarios, there is a need to optimally combine the kernels to perform improved inference. Referred to as Multiple Kernel Learning, this process supports a wide variety of strategies for combining, with the most common choice being the convex combination:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sum_m \beta_m k_m(\mathbf{x}_i, \mathbf{x}_j) \tag{4.2}$$

with $\sum \beta_m = 1$ and $\boldsymbol{\beta} \succeq 0$. In MKL, we optimize for the kernel weights while reducing the empirical risk. The dual formulation for multiple kernel learning can hence be obtained as

$$\begin{aligned} &\min_{\beta} \max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \sum_m \beta_m k_m(\mathbf{x}_i, \mathbf{x}_j) \\ &\text{s.t.} \ \sum_i \alpha_i y_i = 0; 0 \leq \alpha_i \leq C, \forall i; \sum_m \beta_m = 1, \boldsymbol{\beta} \succeq \mathbf{0}. \end{aligned} \tag{4.3}$$

## 4.2 Proposed Approach

In this section, we describe the proposed approach for building an auto-context model in the RKHS, which is comprised of two main steps: (a) constructing the auto-context kernel, (b) integrating image features and the context model using multiple kernel learning.

### 4.2.1 Constructing the Data Kernel

Denote the dataset of $N$ samples belonging to $M$ different classes by $\{(\mathbf{x}^{(n)}, y^{(n)}), n = 1, ..., N\}$, where $\mathbf{x}^{(n)}$ and $y^{(n)} \in \{1, ..., M\}$ are the feature vector and the class label of the image $n$ respectively. For simplicity, we adopt the popular bag-of-words model for building the feature representation. Assuming that there are $S$ diverse descriptors, the visual word dictionaries of sizes $d_1, ..., d_S$ are learned using $k$-means clustering from the extracted descriptors. For a given image $I_n$, its feature representation is $\mathbf{x}^{(n)} = (x_1^{(n)}, ..., x_d^{(n)})$, where $d = \sum_s d_s$ is the feature dimension. Each feature component $x_j^{(n)}$ represents the normalized occurrence frequency of the $j$-th visual word (which is from the $s$-th dictionary) in the image $n$. We then construct the data kernel as the Radial Basis Function (RBF) kernel: $k_F(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}}$, where $\sigma$ is the tuning parameter.

### 4.2.2 Constructing the Auto-Context Kernel

Context information can be defined in different ways based on the specific task in hand. We focus on image classification problem and use the posterior probability map with respect to class labels in the construction of the auto-context kernel. In particular, we begin by estimating the probability map for each image using a kernel SVM classifier in the RKHS. In general, SVM classifiers predict only the class label without providing the probability information explicitly. Given $M$ classes of data, for any example $\mathbf{x}$, the goal is to estimate: $p_i = P(y = i|\mathbf{x}), i \in \{1, 2 \cdots M\}$.

Adopting the one-vs-one classification scheme, we first estimate pairwise class probabilities $r_{ij}$ by fitting a Logistic Regression model (Chang and Lin 2011):

$$r_{ij} = \frac{1}{1 + e^{-\boldsymbol{\gamma}^T \hat{f}}},$$

where $\hat{f}$ is the decision value at $\mathbf{x}$. The parameter $\boldsymbol{\gamma}$ is optimized by minimizing the negative log-likelihood of the training data. Upon estimation of the probabilities for all pairs of classes, we can consider the formulation in (Wu, Lin, and Weng 2003) to estimate the probabilities $p_i$.

$$\min_{\mathbf{p}} \frac{1}{2} \sum_{i=1}^{M} \sum_{j \neq i} (r_{ij} p_i - r_{ij} p_j)^2 \quad \text{s.t. } p_i \geq 0, \sum_{i=1}^{M} p_i = 1. \tag{4.4}$$

This can be efficiently solved by considering its dual problem and using the iterative strategy proposed in (Wu, Lin, and Weng 2003). Given the estimated posterior probabilities, $p(y|\mathbf{x})$, for each image, we build the auto-context kernel following the idea of marginalized kernel (Fernando et al. 2012)

$$\begin{aligned} k_{AC}(\mathbf{x}_i, \mathbf{x}_j) &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j), \\ &= \sum_{y} \sum_{y'} p(y|\mathbf{x}_i, \boldsymbol{\gamma}_y) p(y'|\mathbf{x}_j, \boldsymbol{\gamma}_{y'}) S(y, y'), \end{aligned} \tag{4.5}$$

**Data:** Image feature set $\{(\mathbf{x}^{(n)}, y^{(n)}), n = 1, ..., N\}$, where $y^{(n)} \in \{1, ..., M\}$, $t_{max}$
**Result:** Set of trained classifiers $\{H_t\}_{t=1}^{t_{max}}$
Build image feature kernel $\mathbf{K}_F$ using RBF;
Initialize $p(y|\mathbf{x})$ with uniform distribution and $t = 1$; **while** $t \leq t_{max}$, *i.e., until preset number of iterations is not reached* **do**
1. Construct the auto-context kernel $\mathbf{K}_{AC}^t$ based on the marginal probabilities using (4.5);
2. Perform MKL to obtain the classifier parameters using (4.6) and store them in $H_t$;
3. For each example, calculate the decision function and estimate the probability map $\{p_i\}_{i=1}^M$ by solving (4.4);
4. Set $t \rightarrow t + 1$;

**end**
Return the set of classifiers $\{H_t\}_{t=1}^{t_{max}}$;

**Algorithm 1:** Proposed algorithm for iterative estimation of auto-context in a RKHS setting.

where $S(y, y')$ denotes the similarity between the classes. Note, $k_{AC}(\mathbf{x}_i, \mathbf{x}_j)$ will result in a large similarity when the conditional probabilities that $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to a class $y$ is high. When the weighting term $S(y, y')$ is ignored, (4.5) corresponds to computing the linear kernel for the probability maps. Alternately, we can also construct a RBF kernel for the marginals.

### 4.2.3 Algorithm

With the context information defined inside (4.5), the auto-context kernel measures how close the probability maps are from the ground truth. Consequently, this high-level information can effectively complement the image appearance information in recognition. Integrating the auto-context model into the feature kernel of the observed data is equivalent to fusing the two kernels, and we propose to solve this using multiple kernel learning. Denoting the RKHS corresponding to the image features and context

by $k_F(.,.)$ and $k_{AC}(.,.)$ respectively, the MKL formulation can be written as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \beta_F k_F(\mathbf{x}_i, \mathbf{x}_j) + \beta_{AC} k_{AC}(\mathbf{x}_i, \mathbf{x}_j), \tag{4.6}$$

where $\beta_F, \beta_{AC} \geq 0$ and $\beta_F + \beta_{AC} = 1$. We use the SimpleMKL (Rakotomamonjy et al. 2008) algorithm to obtain the optimal coefficients and classifier parameters. SimpleMKL performs optimization based on gradient descent on the SVM objective through a dual formulation. The overall iterative algorithm is described in Algorithm 1. Initially it is assumed that there is a uniform distribution on $\mathbf{x}$ for all classes, and hence auto-context kernel has no useful information to improve the discrimination. As the algorithm iterates, the auto-context model can be progressively improved by learning the series of kernel classifiers using MKL. In each iteration, the marginal probabilities are estimated in the fused RKHS from the image feature kernel and the auto-context kernel from the previous iteration. Note that after the MKL optimization is completed, there is no need to explicitly calculate the combined kernel result as in (4.6). The marginal probabilities propagate the context information through iterations.

## 4.3 Experiments

In this section, we evaluate the proposed approach using standard visual recognition datasets and study the impact of auto-context modeling. The baseline comparison includes the case of using the feature kernel, and the auto-context kernel independently. Before we present the performance evaluation, we demonstrate the convergence behavior of the proposed algorithm in improving the marginal probabilities using the auto-context model.

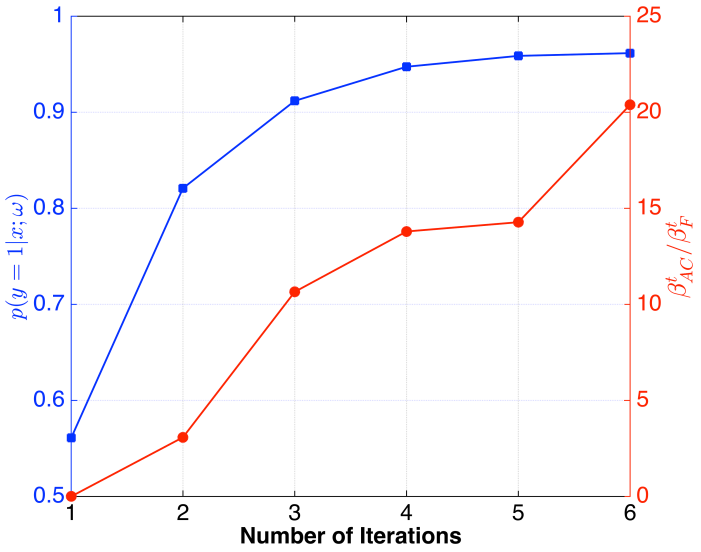



Figure 6: Illustration of the convergence behavior of the proposed algorithm. Left axis shows the conditional probability of an example training sample, with ground truth $y = 1$, estimated by the kernel SVM classifier. Right axis shows the ratio of the importances between the auto-context kernel and the image feature kernel respectively.

#### 4.3.1 Demonstration

The initial context model is equivalent to a uniform probability map with respect to all classes and hence the classification performance solely depends on the feature kernel. As the algorithm progresses, the auto context kernel will attempt to push the class probabilities closer to the ground truth using a series of classifiers. To illustrate this behavior, we consider a binary classification problem using a subset of the Soccer dataset (details in the next section).

Figure 6 illustrates how the relative importance of the auto-context kernel changes over the iterations, with respect to the image feature kernel. More specifically, we consider a training example with ground truth $y = 1$ and analyze the ratio of the weights, $\beta^t_{AC}/\beta^t_F$. In addition, we plot the probability estimate from the kernel SVM, $p(y = 1|\mathbf{x})$. In the first iteration, the uniform context provides no additional information and hence the classifier is solely based on the feature kernel. However,

as our algorithm proceeds, the auto-context kernel enables better discrimination between the two classes and hence $\beta_{AC}$ becomes large. Interestingly, the marginal probability for the sample also changes from 0.55 after iteration 1 to 0.96 after iteration 6 indicating that the auto-context model leads to a more effective classifier.

### 4.3.2 Performance Evaluation

**Soccer Dataset**: This dataset contains images belonging to 7 soccer teams, comprised of 40 images per class. We used 25 images from each class for training and the 15 remaining images for testing. We extracted bag-of-words features based on both the shape and color cues. More specifically, the shape information was characterized by the SIFT descriptors computed at the set of keypoints determined by the Harris-Laplace point detector (Lowe 1999). The Hue-histogram (Van De Weijer and Schmid 2006), which described the color information, was evaluated at the same set of keypoints. Both descriptors are concatenated to construct the image appearance representation. The dictionary sizes for the SIFT and Hue descriptors were fixed at 400 and 300. We constructed the RBF kernel for this image feature set with the parameter $\sigma = 50$. The kernel SVM classifier was designed with the parameter $C = 10$. Furthermore, the cost parameter and $\ell_2$ regularization parameter for multiple kernel learning were set to 15 and 10 respectively. Table 3 shows the performance of our algorithm in comparison to baseline results obtained using only the feature kernel and the one-step auto-context kernel respectively. As the results indicate, the auto-context information enables the marginal probabilities to better match the ground truth in a few iterations, thereby leading to an improved recognition performance.

**UCI Image Segmentation Dataset**: This dataset contains 2310 samples which were drawn randomly from a database of 7 outdoor images. The images were hand-segmented to create a classification for every pixel. Each sample corresponds to a

Table 3: Object recognition performance (% Accuracy) for standard datasets. We compare the performance of our algorithm against that obtained using only the image feature kernel and one step auto-context kernel.

| Dataset | $k_F$+SVM | $k_{AC}$+SVM | Ours |
|---|---|---|---|
| Soccer | 76.2 | 76.2 | **81.9** |
| UCI Segmentation | 86.9 | 85 | **87.9** |

$3 \times 3$ regions. For the classification task, we use the provided 19 different attributes corresponding to the intensity statistics and construct a RBF kernel with $\sigma = 10$. For multiple kernel learning, the $\ell_2$ regularization was fixed at 0.1. As Table 3 indicates, incorporating the auto-context model improves the performance marginally. Note that, this dataset is comparatively easier to classify since the marginal probabilities of the training samples were close to the ground truth even after a single iteration.

Chapter 5

# DEEP KERNEL MACHINE OPTIMIZATION - A PRINCIPLED FRAMEWORK FOR FEATURE FUSION

As discussed in Section 2.2 and 2.3, as two significant representation learning and predictive models in machine learning, kernel methods and deep learning both possess particular advantages and limitations. On one hand, kernel machines and Multiple Kernel Learning are highly effective in learning robust models given even a small set of data. They also generalize to a wide range of problems easily. However, the size of kernel matrices and the complexity of solving the optimization problem scales roughly quadratic with respect to the size of data. This prohibits their applications on large scale datasets. On the other hand, deep learning demonstrated exceptional power in learning multi-level abstractions and flexible complex model. However, in problems characterized by limited dataset sizes and complex dependencies in the input space, deep architecture can get sub-optimal performance due to the requirement of exhaustive tuning of several hyper-parameters.

Inspired by the recent advances which attempt to incorporate ideas from deep learning into kernel machine optimization (see Section 2.3.3), we develop a principled framework to facilitate kernel learning and feature fusion utilize the power of deep architectures. The overall framework is illustrated in Figure 7. Viewed from bottom to top, we first extract multiple dense embeddings from a precomputed kernel matrix $\mathbf{K}$ and optionally the feature domain $\mathcal{X}$ 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 another layer

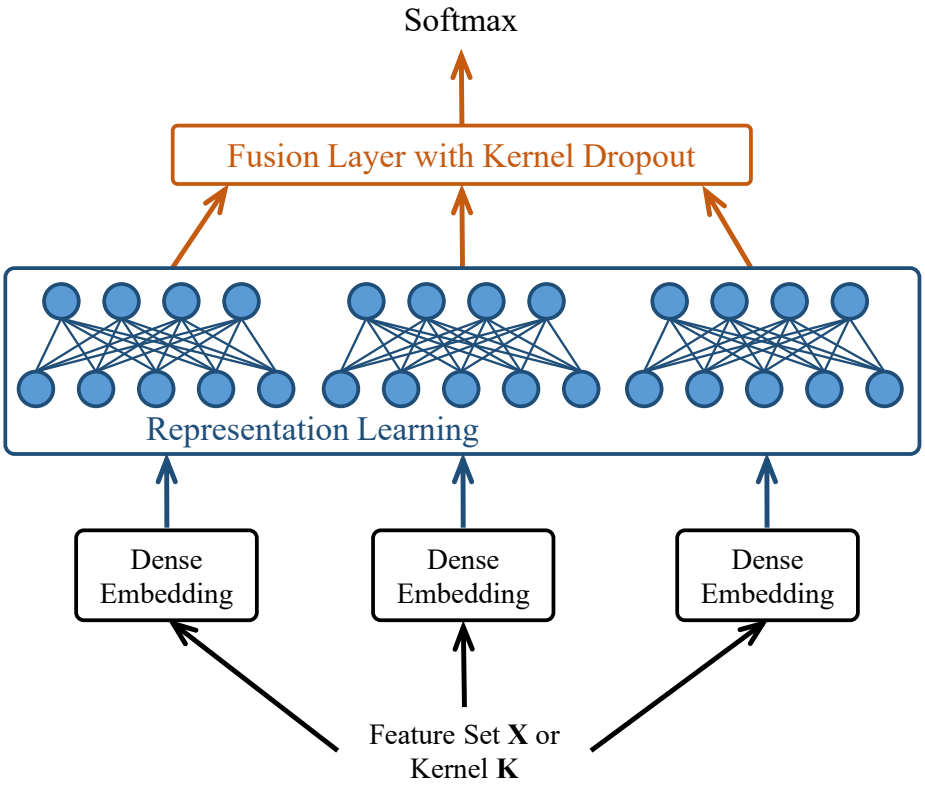



Figure 7: *DKMO* - Proposed approach for optimizing kernel machines using deep neural networks. For a given kernel, we generate multiple dense embeddings using kernel approximation techniques, and fuse them in a fully connected deep neural network. The architecture utilizes fully connected networks with 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.

which is responsible for combining them 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 multiple 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.

## 5.1 Background on Kernel Approximation

Broadly speaking, there are two class of approaches commonly used by researchers to alleviate the scalability issue of kernel methods. First, kernel approximation strategies can be used to reduce both computational and memory complexity of kernel methods, e.g. the Nyström method (Drineas and Mahoney 2005). The crucial component in Nyström method is to select a subset of the kernel matrix for approximation. Straightforward uniform sampling has been demonstrated to provide reasonable performance in many cases (Kumar, Mohri, and Talwalkar 2012). In (Zhang and Kwok 2010), the authors proposed an improved variant of Nyström approximation, that employs $k$-means 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. In (Kumar, Mohri, and Talwalkar 2009), Kumar *et. al.* generate 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 (Rahimi and Recht 2008; Le, Sarlos, and Smola 2013). Instead of using the implicit feature mapping in the kernel trick, the authors in (Rahimi and Recht 2008) proposed a random feature method 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.* (Huang et al. 2014) showed that shallow kernel machines matched the performance of deep networks in speech recognition, while being computationally efficient.

Consider the kernel Gram matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, where $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. Operating

with kernel matrices makes kernel methods highly ineffective in large-scale problems. Consequently, there is a need to significantly reduce the computational and memory complexity for scaling kernel methods. In kernel approximation, the objective is 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 $r \ll n$. Truncated Singular Value Decomposition (SVD) factorizes $\mathbf{K}$ to $\mathbf{U_K}\mathbf{\Lambda_K}\mathbf{U_K}^T$, where $\mathbf{\Lambda_K} = \text{diag}(\sigma_1, \ldots, \sigma_n)$ contains the eigenvalues in non-increasing order and $\mathbf{U_K}$ contains the corresponding eigenvectors. Subsequently, a rank-$r$ approximation $\tilde{\mathbf{K}}_r$ is constructed using the top eigenvectors, i.e. $\tilde{\mathbf{K}}_r = \sum_{i=1}^{r} \sigma_i^{-1} \mathbf{U}_\mathbf{K}^{(i)} {\mathbf{U}_\mathbf{K}^{(i)}}^T$. This procedure provides the optimal rank-$r$ approximation in terms of the Frobenius norm; however this incurs $O(n^3)$ time complexity making it infeasible in practice. While several kernel approximation methods exist, Nyström methods outperform other existing greedy and random sampling approaches (Drineas and Mahoney 2005) and hence we choose them as an important building block.

In the Nyström method, a subset of $s$ columns are selected from $\mathbf{K}$ to approximate the eigen-system of the kernel matrix. Denote $\mathbf{W} \in \mathbb{R}^{s \times s}$ as the intersection of the selected columns and corresponding rows on $\mathbf{K}$ and $\mathbf{E} \in \mathbb{R}^{n \times s}$ containing the selected columns. The rank-$r$ approximation $\tilde{\mathbf{K}}_r$ of $\mathbf{K}$ is computed as:

$$\tilde{\mathbf{K}}_r = \mathbf{E}\tilde{\mathbf{W}}_r\mathbf{E}^T \tag{5.1}$$

where $r \leq s$ and $\tilde{\mathbf{W}}_r$ is the optimal rank-$r$ approximation of $\mathbf{W}$ obtained using truncated SVD. As 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 (M. Li et al. 2015). The approximate mapping function $\mathbf{L}$ can then be obtained by

$$\mathbf{L} = \mathbf{E}(\mathbf{U}_{\tilde{\mathbf{W}}_r}\mathbf{\Lambda}_{\tilde{\mathbf{W}}_r}^{-1/2}) \tag{5.2}$$

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

## 5.2 Deep Kernel Machine Optimization

### 5.2.1 Dense Embedding Layer

From Figure 7 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 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. **Motivation:** onsider 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. As shown in Section, this procedure can be viewed as kernel approximation with truncated SVD or Nyström methods. Furthermore, this is conceptually similar to the process of obtaining dense word embeddings in natural language processing. For example, Levy *et.al* (Levy and Goldberg 2014) have showed 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 (Levy and Goldberg 2014).

In existing deep kernel learning approaches such as the convolutional kernel networks (Mairal 2016), the key idea is to construct multiple reproducing kernel Hilbert spaces at different layers of the network, with a sequence of pooling operations between the layers to facilitate kernel design for different sub-region 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 from the feature set or the kernel matrix using Nyström methods, and utilize the DNN as both representation learning and feature fusion mechanisms to obtain an explicit representation for data in the Euclidean space.

**Dense Embeddings using Nyström Approximation:** In order to be flexible with different application-specific constraints, we consider two different pipelines for constructing the dense embeddings based on Nyström approximation: I) When the input data is constructed from pre-defined feature sources, we employ the clustered Nyström method (Zhang and Kwok 2010), which identifies a subspace in the RKHS using clustering algorithms, and explicitly project the feature mappings in RKHS onto the subspace. In this case, the dense embeddings can be obtained without constructing the complete kernel matrix for the dataset. II) In many applications involving DNA sequences and graphs, obtaining the kernel matrices is often easier than extracting effective features for inference tasks. Furthermore, for many existing datasets, pair-wise distance matrices are already formed and can be easily converted into kernel matrices. In such scenarios, we use the conventional Nyström method shown in Section 5.1 to calculate the dense embeddings. Next, we discuss in detail the two strategies:

1. **Clustered Nyström approximations on feature set:** Following the approach in (Zhang and Kwok 2010), $k$-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 Appendix of (Mairal 2016) for detailed derivation):

$$\mathbf{L}_\mathbf{Z} = \mathbf{E}_\mathbf{Z}\mathbf{W}_\mathbf{Z}^{-1/2}. \tag{5.3}$$

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 Equations (5.3) and (5.2) one can notice that $\mathbf{L}_\mathbf{Z}$ is directly related to $\mathbf{L}$ by a linear transformation when $r = s$, since

$$\mathbf{W}_\mathbf{Z}^{-1/2} = \mathbf{U}_\mathbf{Z}\mathbf{\Lambda}_\mathbf{Z}^{-1/2}\mathbf{U}_\mathbf{Z}^T, \tag{5.4}$$

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

With different sets of clustering centroids spanning distinct subspaces $\{\mathcal{F}_i\}$, the projections will result in completely different representations. 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. 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 landmark selection process with different

clusterings techniques, such as the $k$-means, $k$-medians, $k$-medoids, agglomerative clustering (Kaufman and Rousseeuw 2009) and spectral clustering based on $k$ nearest neighbors (Von Luxburg 2007). 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 means of the features in that cluster. In summary, based on $P$ 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 Equation (5.3).

2. **Conventional Nyström approximations on kernel:** In contrast to the previous case, in applications where the feature sources are not directly accessible, we need to construct dense embeddings from the kernel matrix. In order to achieve this, we extract an ensemble of kernel approximate mappings through different random sampling sets of the kernel matrix. From $\mathbf{K}$, we randomly select $s \times P$ columns without replacement, and then divide it into $P$ sets containing $s$ columns each. The resulting matrices $\mathbf{W}_1, \ldots, \mathbf{W}_P$, along with the matrices $\mathbf{E}_1, \ldots, \mathbf{E}_P$ defined in Section 5.1, provide the dense embeddings $\mathbf{L}_1, \ldots, \mathbf{L}_P$ following Equation (5.2). This is conceptually similar to (Kumar, Mohri, and Talwalkar 2009), 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 instead of kernels and the mappings are further coupled with the task-specific optimization enabled by the deep architecture.

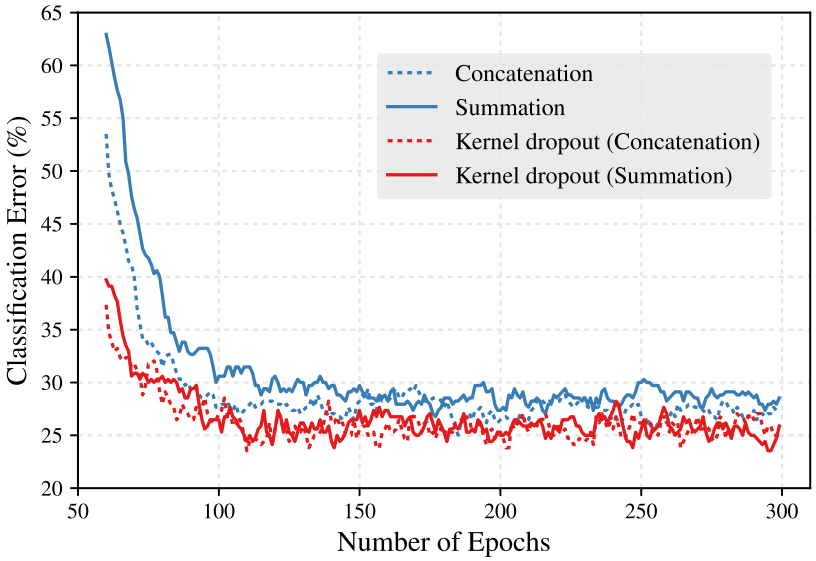



Figure 8: Effects of kernel dropout on the DKMO training process: We compare the convergence characteristics obtained with the inclusion of the kernel dropout regularization in the fusion layer in comparison to the non-regularized version. Note, we show the results obtained with two different merging strategies - concatenation and summation. We observe that the kernel dropout regularization leads to improved convergence and lower classification error for both the merging styles.

### 5.2.2 Representation Learning

Given the kernel-specific dense embeddings, for each embedding, we perform representation learning using a multi-layer fully connected network to obtain more concise representation for subsequent fusion and prediction stages. 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 (Srivastava et al. 2014) is used to prevent overfitting and batch normalization (Ioffe and Szegedy 2015) is adopted to accelerate training.

### 5.2.3 Fusion Layer with Kernel Dropout

The fusion layer receives the latent representations for each of the kernel approximated 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 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 (Srivastava et al. 2014) 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 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:

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

where $\mathbf{w}_i$ are the weights for hidden unit $i$, $(\cdot)$ denotes vector concatenation and $f$ is the softmax activation function. In Figure 8, 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 5.4.2). 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.

## 5.3 M-DKMO: Extension to Multiple Kernel Learning

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 deep kernel machine optimization (*M-DKMO*) presented in the previous section.

In order to optimize kernel machines with multiple kernels $\{\mathbf{K}\}_{m=1}^{M}$ (optionally

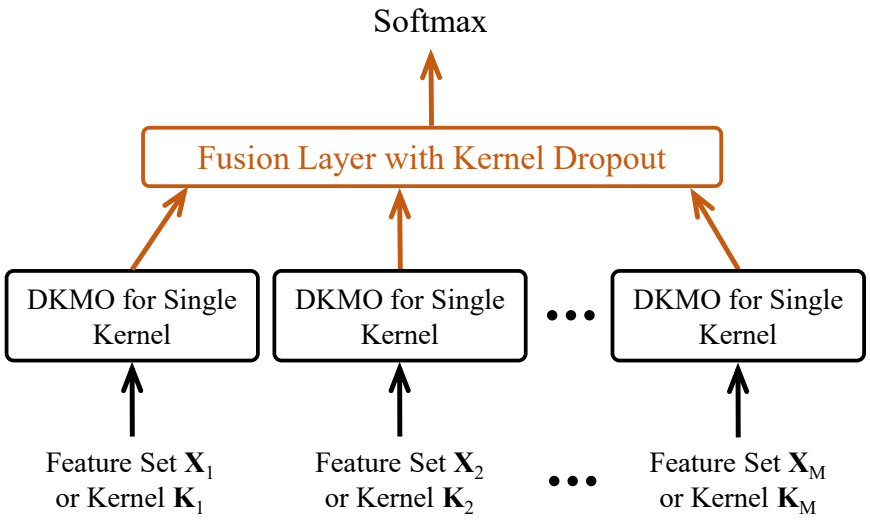



Figure 9: *M-DKMO* - Extending the proposed deep kernel optimization approach to the case of multiple kernels. Each of the kernels are first independently trained with the *DKMO* algorithm and then combined using a global fusion layer. The parameters of the global fusion layer and the individual *DKMO* networks are fine-tuned in an end-to-end learning fashion.

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. Figure illustrates the *M-DKMO* algorithm for multiple kernel learning. As shown in the figure, an end-to-end learning network is constructed based on a set of pre-trained 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 Figure 7, the global fusion layer can admit any merging strategy and can optionally include additional fully connected layers before the softmax layer.

Note that, after pre-training 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 Figure. Furthermore, we adopt the kernel dropout strategy described in 5.2.3 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.

## 5.4 Experimental Results

In this section, we demonstrate the features and performance of the proposed framework using 3-fold experiments on real-world datasets:

1. In Section 5.4.1, 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 datasets with pre-computed features. A sample set of images from this dataset are shown in Figure 10(a).
2. In Section 5.4.2, we emphasize the effectiveness of proposed architectures when only pair-wise 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 Figure 10(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 5.4.3, we focus on the performance of the proposed architecture when limited training data is available. As a representative application, sensor

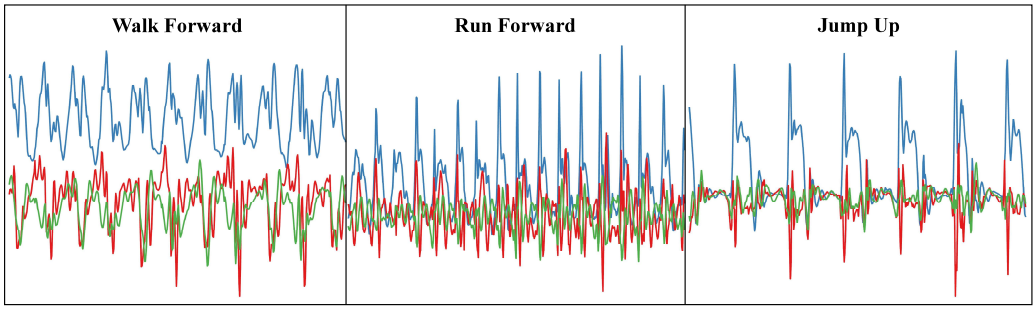
(a) Images from different classes in the flowers102 dataset

| **Cytosolic and Nuclear Sequences** |
|---|
| MVTPALQMKKPKQFCRRMGQKKQRPARAGQPHS… |
| MPARGGSARPGRGSLKPVSVTLLPDTEQPPFLGRA… |
| MFLEVADLKDGLWVWKVVFLQVCIEASGWGAEV… |

| **Mitochondrion Sequences** |
|---|
| MLRATLARLEMAPKVTHIQEKLLINGKFVPAVSGK… |
| MLRAALSTARRGPRLSRLLSAAATSAVPAPNQQPE… |
| MYRRLGEVLLLSRAGPAALGSAAADSAALLGWAR… |

| **Secretory Pathway Sequences** |
|---|
| MVEMLPTAILLVLAVSVVAKDNATCDGPCGLRFRQNPQG… |
| MQLLRCFSIFSVIASVLAQELTTICEQIPSPTLESTPYSLST… |
| MIQGLESIMNQGTKRILLAATLAATPWQVYGSIEQPSLLP… |

(b) Sequences belonging to 3 different classes in the non-plant dataset for protein subcellular localization




(c) Accelerometer measurements characterizing different activities from the USC-HAD dataset

Figure 10: Example samples from the datasets 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.

based activity recognition requires often laborious data acquisition from human subjects. The difficulty in obtaining large amounts of clean and labeled data can

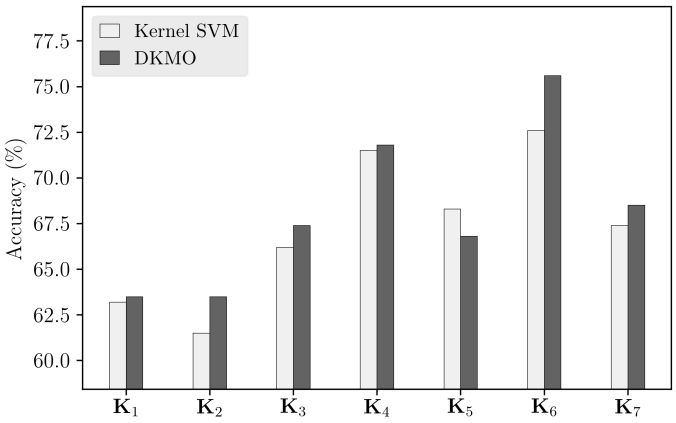
be further complicated by sensor failure, human error and incomplete coverage of the demographic diversity (Kwapisz, Weiss, and Moore 2011; Zhang and A. A. Sawchuk 2012; Song, Thiagarajan, Ramamurthy, Spanias, and Turaga 2016). Therefore, it is significant to have a model which has strong extrapolation ability given even very limited training data. When features are accessible, an alternative general-purpose algorithm is fully connected neural networks (*FCN*) 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 Figure 10(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 pre-processing 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 5.2.1, the proposed DKMO algorithm can handle both these scenarios by constructing the dense embeddings suitably.

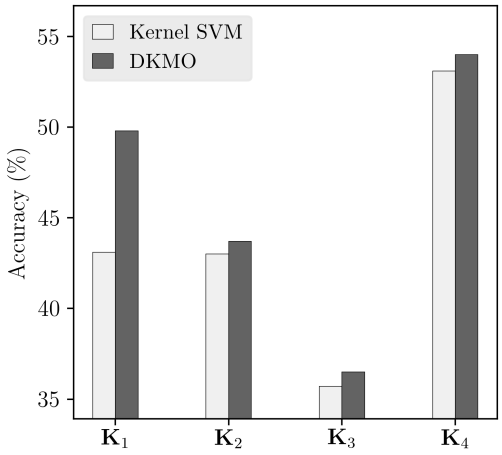
We summarize all methods used in our comparative studies and the details of the parameters used in our experiments below:

*Kernel SVM.* A single kernel SVM is applied on each of the kernels. Following (Chapelle and Zien 2005), 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 $C^{*} = 1/(\frac{1}{n}\sum_{i}\mathbf{K}_{i,i} - \frac{1}{n^{2}}\sum_{ij}\mathbf{K}_{i,j})$, which is the inverse of the empirical variance of data in the input space.

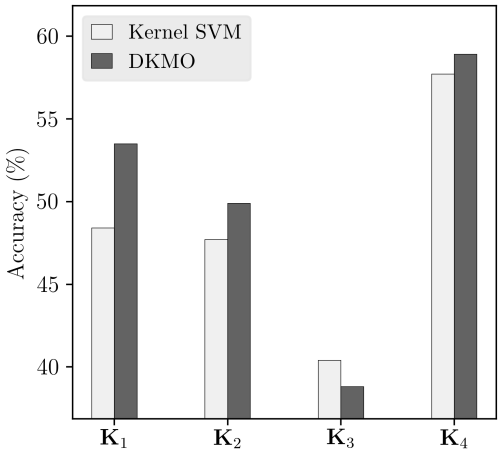
*Uniform.* Simple averaging of base kernels has been shown to be a strong baseline




(a) Flowers17




(b) Flowers102 - 20




(c) Flowers102 - 30

Figure 11: Single Kernel Performance on Flowers Datasets

in comparison to MKL (Gehler and Nowozin 2009; Cortes, Mohri, and Rostamizadeh 2009). We then apply *kernel SVM* on the averaged kernel.

*UFO-MKL.* We compare with this state-of-the-art multiple kernel learning algorithm (Orabona and Jie 2011). 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 5.4.2), 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 4 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 datasets, we first applied the DKMO approach to each of the kernels (as in Figure 7) with the same network size as in *FCN*. Based on the discussion in Section 5.2.1, for datasets that allow access to explicit feature sources, we extracted 5 dense embeddings corresponding to the 5 landmark point sets obtained using different clustering algorithms. On the other hand, for datasets with only kernel similarity matrices between the samples, we constructed 6 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 $\bar{\mathbf{K}}_{i,j} = \mathbf{K}_{i,j}/\sqrt{\mathbf{K}_{i,i}\mathbf{K}_{j,j}}$, 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 same as the ones in the *FCN* method. All network architectures were implemented using the Keras library (Chollet 2015) with the TensorFlow backend and trained on a single GTX 1070 GPU.

### 5.4.1 Image Classification - Comparisons with Kernel Optimization and Multiple Kernel Learning

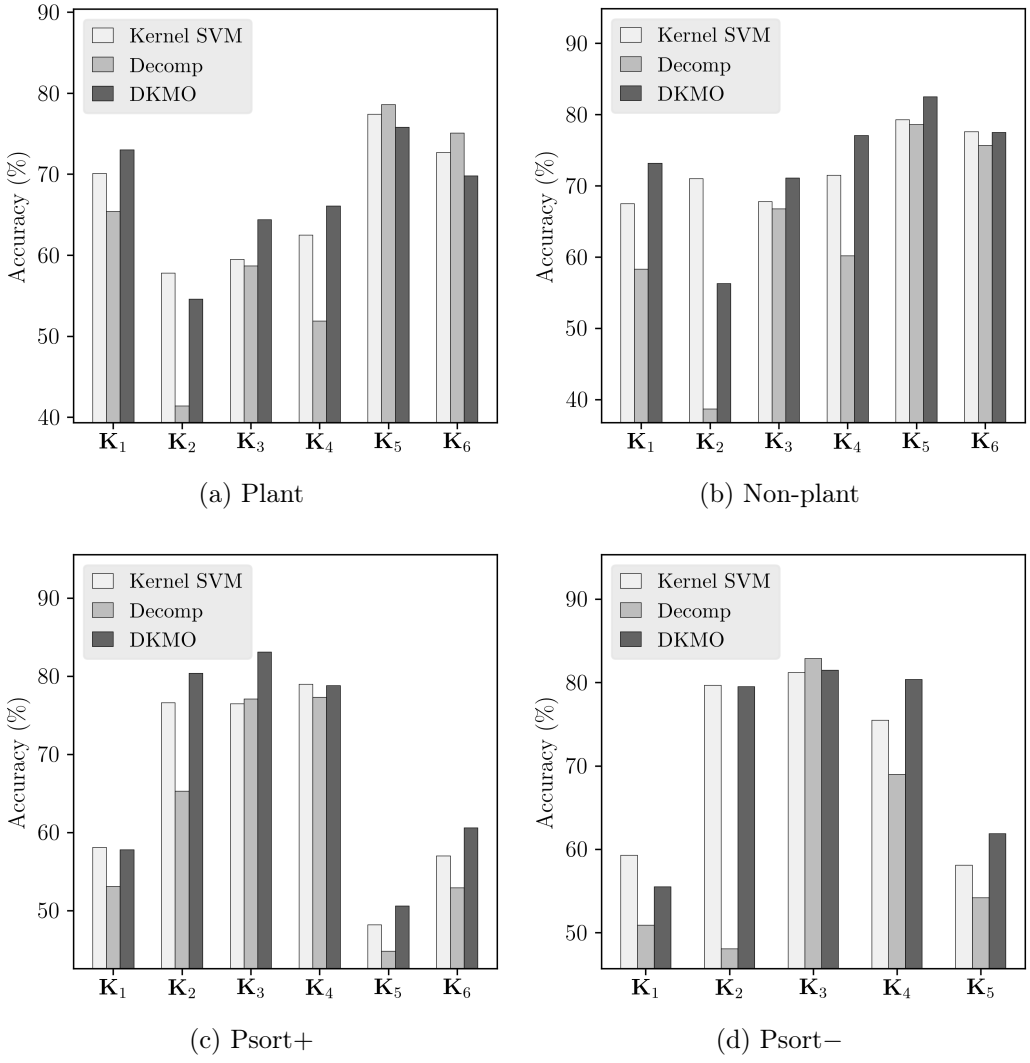
In this section, we consider the performance of the proposed approach in image classification tasks, using datasets which have had proven success with kernel methods. More specifically, we compare *DKMO* with *kernel SVM*, and *M-DKMO* with *Uniform* and *UFO-MKL* respectively to demonstrate that one 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 datasets for image classification with kernel methods. Both datasets are comprised of flower images belonging to 17 and 102 categories respectively. The precomputed $\chi^2$ distance matrices were calculated based on bag of visual words of features such as HOG, HSV, SIFT etc. 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 (Jhuo and Lee 2010; Thiagarajan et al. 2014; Bucak, Jin, and Jain 2014; Karthikeyan Ramamurthy et al. 2016).

We construct $\chi^2$ 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 (Orabona, Jie, and Caputo 2012), the $\gamma$ value is empirically estimated as the inverse of the average pairwise distances. To be consistent with the setting from (Qi et al. 2014) on the flowers102 dataset, we consider training on both 20 samples per class and 30 samples per class respectively. The experimental results for single kernels are shown in Figure 11 and results for multiple kernel fusion are shown in Table 4, where we measure the

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

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 dataset, kernel 1 in the flowers102 dataset and the multiple kernel fusion result for the flowers17 dataset.

### 5.4.2 Protein Subcellular Localization - Lack of Explicit Feature Sources




(a) Plant

(b) Non-plant

(c) Psort+

(d) Psort−

Figure 12: Single Kernel Performance on Protein Subcellular Datasets

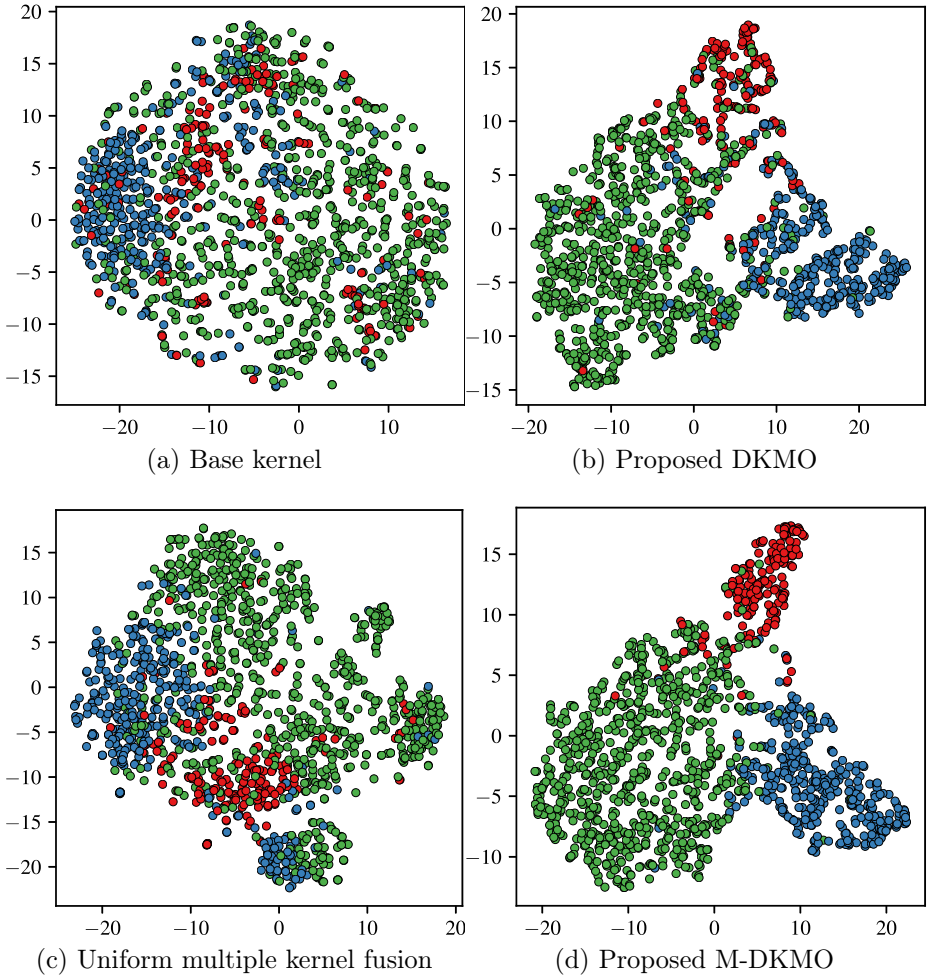
Table 4: Multiple Kernel Fusion Performance on Flowers Datasets

| **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** |

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 (Ong and Zien 2008; Ding and Dubchak 2001; Andreeva et al. 2014). More specifically, we focus on predicting the protein subcellular localization from protein sequences. We use 4 datasets from (Ong and Zien 2008) [2]: plant, non-plant, psort+ and psort− belonging to $3-5$ classes. Among the 69 sequence motif kernels, we sub-select 6, which encompass all 5 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 Figure 12 and Table 5. The first observation is that for *Decomp*, although the optimal 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 datasets, *DKMO* consistently produces improved or at least similar classification accuracies in

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



(a) Base kernel (b) Proposed DKMO

(c) Uniform multiple kernel fusion (d) Proposed M-DKMO

Figure 13: 2D T-SNE visualizations of the representations obtained for the non-plant dataset using the base kernel (Kernel 5), uniform multiple kernel fusion, and the learned representations from DKMO and M-DKMO approaches. The samples are colored by their corresponding class associations.

comparison to the baseline *kernel SVM*. For the few cases where *DKMO* is inferior, for example kernel 2 in non-plant, 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 multiple kernel learning case, the proposed *M-DKMO* approach produces improved performance consistently.

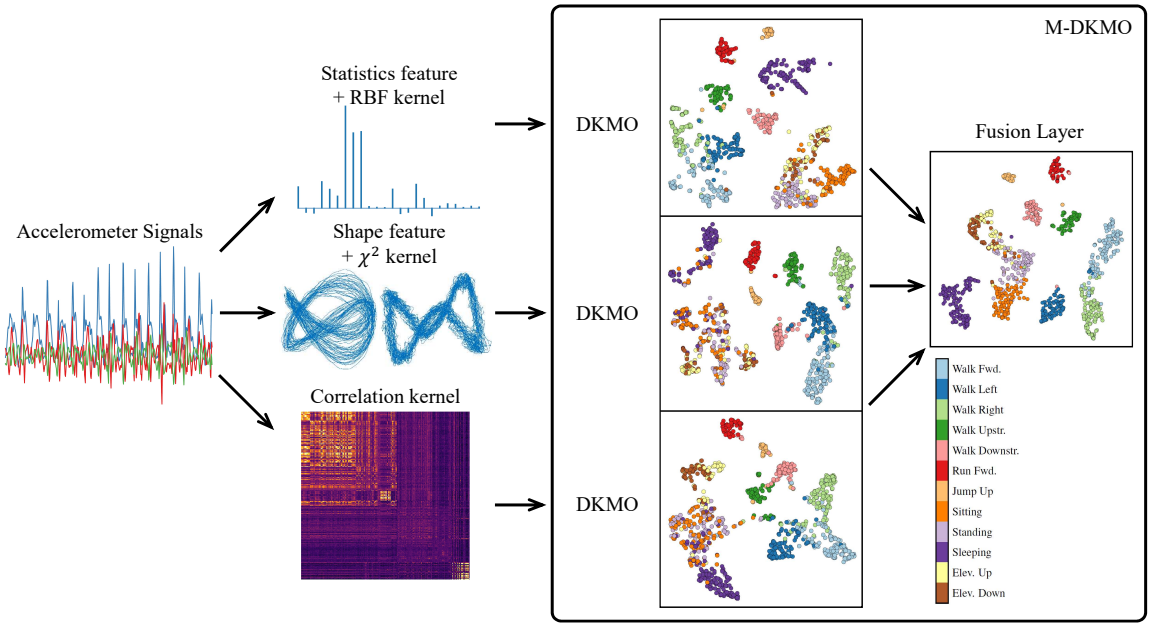



Figure 14: Visualization of proposed framework applied on USD-HAD dataset: we show the raw 3-axis accelerometer signal and extracted 3 distinct types of features: the time-series statistics, topological structure where we extract TDE descriptors and the correlation kernel. Furthermore, we show the t-SNE visualization of the representations learned by *DKMO* and *M-DKMO*, where all points are classes coded according to the colorbar.

Finally, in order to understand the behavior of the representations generated by different approaches, we employ the t-SNE algorithm (Maaten and Hinton 2008) and obtain 2-D visualizations of the considered baselines and the proposed approaches (Figure 24). For demonstration, we consider the representation from *Decomp* of kernel 5 and *Decomp* of the kernel from *Uniform* in the non-plant dataset. In both *DKMO* and *M-DKMO*, we performed t-SNE on the representation obtained from the fusion layers. The comparisons in the Figure 24 show that the proposed single kernel learning and kernel fusion methods produce highly discriminative representations than the corresponding conventional approaches.

Table 5: Multiple Kernel Fusion Performance on Protein Subcellular Datasets

| **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 |

### 5.4.3 Sensor-based Activity Recognition - Limited Data Case

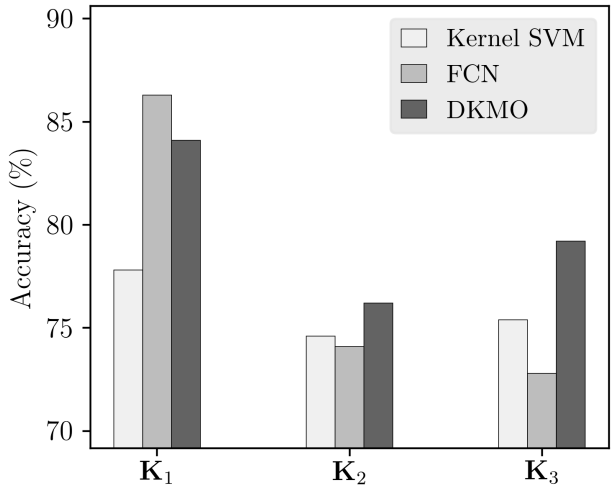



Figure 15: Single Kernel Performance on USC-HAD Datasets

In this section, we focus on evaluating the performance of 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.

Recent advances in activity recognition have demonstrated promising results in fitness monitoring and assisted living (Zhang and A. A. Sawchuk 2012; S. Zhang et al. 2016, 2018). However, when applied to smartphone sensors and wearables, existing algorithms still have limitations dealing with the measurement inaccuracies and noise. In (Song, Thiagarajan, Ramamurthy, Spanias, and Turaga 2016), the authors proposed to address this challenge by performing sensor fusion, wherein each sensor is characterized by multiple feature sources, which naturally enables multiple kernel learning schemes.

We evaluate the performance of our framework using the USC-HAD dataset [3], which contains 12 different daily activities performed by each of the subjects. The measurements are obtained using a 3-axis accelerometer at a sampling rate of 100Hz. Following the standard experiment methodology, we extract non-overlapping frames of 5 seconds each, creating a total of 5353 frames. We perform a $80-20$ 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 3 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 (TDE) to model the underlying dynamical system (Frank, Mannor, and Precup 2010). The TDEs of a time-series signal $\mathbf{x}$ can be defined as a matrix $\mathbf{S}$ whose $i$th row is $\mathbf{s}_i = [x_i, x_{i+\tau}, \ldots, x_{t+(d'-1)\tau}]$, where $d'$ is number of samples and $\tau$ is the de-

[3] sipi.usc.edu/HAD

Table 6: Multiple Kernel Fusion Performance on USC-HAD Datasets

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

lay 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 $\tau$ is fixed to 10 and embedding dimension $d'$ is chosen to be 8. Following the approach in (Frank, Mannor, and Precup 2010), we use Principle Component Analysis (PCA) to project the embedding to 3-D for noise reduction. To model the topology of the delayed observations in 3-D, we measure the pair-wise distances between samples as $\|\mathbf{s}_i - \mathbf{s}_j\|_2$ (Venkataraman and Turaga 2016) and build the distance histogram feature with a pre-specified 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 $\mathbf{R} = \mathbf{U_R}\mathbf{\Lambda_R}\mathbf{U_R}^T$, where $\mathbf{\Lambda_R} = \text{diag}(\sigma_1, \ldots, \sigma_n)$ and $\sigma_1 \geq \cdots \geq \sigma_r \geq 0 \geq \sigma_{r+1} \geq ... \geq \sigma_n$, the correlation kernel is constructed as $\mathbf{K} = \mathbf{U_R}\hat{\mathbf{\Lambda}}_\mathbf{R}\mathbf{U_R}^T$, where $\hat{\mathbf{\Lambda}}_\mathbf{R} = \text{diag}(\sigma_1, \ldots, \sigma_r, 0, ..., 0)$.

Figure 14 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 $\chi^2$ 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 Figure 14, 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 Figure 15, 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 6, we have striking observations: 1) For *FCN*, the fusion performance is in fact dragged down by the poor performance on shape and correlation features as in Figure 15. 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.

# Chapter 6

# MULTIVARIATE TIME SEQUENCE ANALYSIS USING ATTENTION MODELS

## 6.1 Multivariate clinical data analysis

Healthcare is one of the prominent applications of data mining and machine learning, and it has witnessed tremendous growth in research interest recently. This can be directly attributed to both the abundance of digital clinical data, primarily due to the widespread adoption of electronic health records (EHR), and advances in data-driven inferencing methodologies. Clinical data, for example intensive care unit (ICU) measurements, is often comprised of multi-variate, time-series observations corresponding to sensor measurements, test results and subjective assessments. Potential inferencing tasks using such data include classifying diagnoses accurately, estimating length of stay, and predicting future illness, or mortality.

The classical approach for healthcare data analysis has been centered around extracting hand-engineered features and building task-specific predictive models. Machine learning models are often challenged by factors such as need for long-term dependencies, irregular sampling and missing values. In the recent years, recurrent Neural Networks (RNNs) based on Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) have become the de facto solution to deal with clinical time-series data. RNNs are designed to model varying-length data and have achieved state-of-the-art results in sequence-to-sequence modeling (Sutskever, Vinyals, and Le 2014), image captioning (Xu et al. 2015) and recently in clinical diagnosis (Lipton et al. 2015). Furthermore, LSTMs are effective in exploiting long-range dependencies and handling nonlinear dynamics.

RNNs perform computations at each position of the time-series by generating a sequence of hidden states as a function of the previous hidden state and the input for current position. This inherent sequential nature makes parallelization challenging. Though efforts to improve the computational efficiency of sequential modeling have recently surfaced, some of the limitations still persist. The recent work of Vaswani *et. al.* argues that attention mechanisms, without any recurrence, can be effective in sequence-to-sequence modeling tasks. Attention mechanisms are used to model dependencies in sequences without regard for their actual distances in the sequence (Bahdanau, Cho, and Bengio 2014).

Another important factor that has challenged machine learning research towards clinical diagnosis is the lack of universally accepted benchmarks to rigorously evaluate the modeling techniques. Consequently, in an effort to standardize research in this field, in (Harutyunyan et al. 2017), the authors proposed public benchmarks for four different clinical tasks: mortality prediction, detection of physiologic decompensation, forecasting length of stay, and phenotyping. Interestingly, these benchmarks are supported by the Medical Information Mart for Intensive Care (MIMIC-III) database (Johnson et al. 2016), the largest publicly available repository of rich clinical data currently available. These datasets exhibit characteristics that are typical of any large-scale clinical data, including varying-length sequences, skewed distributions and missing values. In (Lipton et al. 2015; Harutyunyan et al. 2017), the authors established that RNNs with LSTM cells outperformed all existing baselines including methods with engineered features.

In this section, we develop *SAnD* (Simply Attend and Diagnose), a new approach for clinical time-series analysis, which is solely based on attention mechanisms. In contrast to sequence-to-sequence modeling in NLP, we propose to use *self-attention*

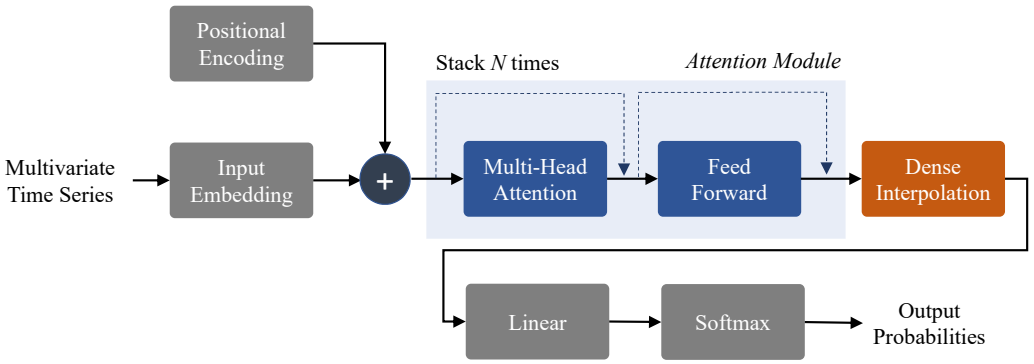



Figure 16: An overview of the proposed approach for clinical time-series analysis. In contrast to state-of-the-art approaches, this does not utilize any recurrence or convolutions for sequence modeling. Instead, it employs a simple self-attention mechanism coupled with a dense interpolation strategy to enable sequence modeling. The attention module is comprised of $N$ identical layers, which in turn contain the attention mechanism and a feed-forward sub-layer, along with residue connections.

that models dependencies within a single sequence. In particular, we adopt the *multi-head* attention mechanism similar to (Vaswani et al. 2017), with an additional masking to enable causality. In order to incorporate temporal order into the representation learning, we propose to utilize both positional encoding and a dense interpolation embedding technique. Our evaluation of *SAnD* on all MIMIC-III benchmark tasks and show that it is highly competitive, and in most cases outperforms the state-of-the-art LSTM based RNNs. Both superior performance and computational efficiency clearly demonstrate the importance of attention mechanisms in clinical data.

### 6.1.1 Proposed Approach

The effectiveness of LSTMs have been established in a wide-range of clinical prediction tasks. In here, we are interested in studying the efficacy of attention models in similar problems, dispensing recurrence entirely. While core components

from the Transformer model (Vaswani et al. 2017) can be adopted, key architectural modifications are needed to solve multivariate time-series inference problems.

The motivation for using attention models in clinical modeling is three-fold: (i) *Memory*: While LSTMs are effective in sequence modeling, lengths of clinical sequences are often very long and in many cases they rely solely on short-term memory to make predictions. Attention mechanisms will enable us to understand the amount of memory modeling needed in benchmark tasks for medical data; (ii) *Optimization*: The mathematical simplicity of attention models will enable the use of additional constraints, e.g. explicit modeling of correlations between different measurements in data, through inter-attention; (iii) *Computation*: Parallelization of sequence model training is challenging, while attention models are fully parallelizable.

Our architecture is inspired by the recent Transformer model for sequence transduction (Vaswani et al. 2017), where the encoder and decoder modules were comprised solely of an attention mechanism. The Transformer architecture achieves superior performance on machine translation benchmarks, while being significantly faster in training when compared to LSTM-based recurrent networks (Sutskever, Vinyals, and Le 2014; Wu et al. 2016). Given a sequence of symbol representations (e.g. words) $(x_1, \ldots, x_T)$, the encoder transforms them into a continuous representation $\mathbf{z}$ and then the decoder produces the output sequence $(y_1, \ldots, y_T)$ of symbols.

Given a sequence of clinical measurements $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$, $\mathbf{x}_t \in \mathbb{R}^R$ where $R$ denotes the number of variables, our objective is to generate a sequence-level prediction. The type of prediction depends on the specific task and can be denoted as a discrete scalar $y$ for multi-class classification, a discrete vector $\mathbf{y}$ for multi-label classification and a continuous value $y$ for regression problems. The proposed architecture is shown in Figure 16. In the rest of this section, we describe each of the components in detail.

**Input Embedding:** Given the $R$ measurements at every time step $t$, the first step in our architecture is to generate an embedding that captures the dependencies across different variables without considering the temporal information. This is conceptually similar to the input embedding step in most NLP architectures, where the words in a sentence are mapped into a high-dimensional vector space to facilitate the actual sequence modeling (Kim 2014). To this end, we employ a 1D convolutional layer to obtain the $d$-dimensional ($d > R$) embeddings for each $t$. Denoting the convolution filter coefficients as $\mathbf{w} \in \mathbb{R}^{T \times h}$, where $h$ is the kernel size, we obtain the input embedding: $\mathbf{w} \cdot \mathbf{x}_{i:i+h-1}$ for the measurement position $i$.

**Positional Encoding:** Since our architecture contains no recurrence, in order to incorporate information about the order of the sequence, we include information about the relative or absolute position of the time-steps in the sequence. In particular, we add *positional encodings* to the input embeddings of the sequence. The encoding is performed by mapping time step $t$ to the same randomized lookup table during both training and prediction. The $d$-dimensional positional embedding is then added to the input embedding with the same dimension. Note that, there are alternative approaches to positional encoding, including the sinusoidal functions in (Vaswani et al. 2017). However, the proposed strategy is highly effective in all our tasks.

**Attention Module:** Unlike transduction tasks in NLP, our inferencing tasks often require classification or regression architectures. Consequently, *SAnD* relies almost entirely on self-attention mechanisms. Self-attention, also referred as intra-attention, is designed to capture dependencies of a single sequence. Self-attention has been used successfully in a variety of NLP tasks including reading comprehension (Cui et al. 2016) and abstractive summarization (Paulus, Xiong, and Socher 2017). As we will describe later, we utilize a restricted self-attention that imposes causality, i.e., considers

information only from positions earlier than the current position being analyzed. In addition, depending on the task we also determine the range of dependency to consider. For example, we will show in our experiments that phenotyping tasks require a longer range dependency compared to mortality prediction.

In general, an attention function can be defined as mapping a query $\mathbf{q}$ and a set of key-value pairs $\{\mathbf{k}, \mathbf{v}\}$ to an output $\mathbf{o}$. For each position $t$, we compute the attention weighting as the inner product between $\mathbf{q}_t$ and keys at every other position in the sequence (within the restricted set) $\{\mathbf{k}_{t'}\}_{t'=t-r}^{t-1}$, where $r$ is the mask size. Using these attention weights, we compute $\mathbf{o}$ as weighted combination of the value vectors $\{\mathbf{v}_{t'}\}_{t'=t-r}^{t-1}$ and pass $\mathbf{o}$ through a feed-forward network to obtain the vector representation for $t$. Mathematically, the attention computation can be expressed as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathbf{T}}}{\sqrt{d}}\right)\mathbf{V}, \tag{6.1}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are the matrices formed by query, key and value vectors respectively, and $d$ is the dimension of the key vectors. This mechanism is often referred to as the scalar dot-product attention. Since we use only self-attention, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ all correspond to input embeddings of the sequence (with position encoding). Additionally, we mask the sequence to specify how far the attention models can look into the past for obtaining the representation for each position. Hence, to be precise, we refer to this as *masked self-attention*.

Implicitly, self-attention creates a graph structure for the sequence, where edges indicate the temporal dependencies. Instead of computing a single attention graph, we can actually create multiple attention graphs each of which is defined by different parameters. Each of these attention graphs can be interpreted to encode different types of edges and hence can provide complementary information about different

types of dependencies. Hence, we use "multi-head attention" similar to (Vaswani et al. 2017), where 8 heads are used to create multiple attention graphs and the resulting weighted representations are concatenated and linearly projected to obtain the final representation. The second component in the attention module is 1D convolutional sub-layers with kernel size 1, similar to the input embedding. Internally, we use two of these 1D convolutional sub-layers with ReLU (rectified linear unit) activation in between. Note that, we include residue connections in both the sub-layers.

Since we stack the attention module $N$ times, we perform the actual prediction task using representations obtained at the final attention module. Unlike transduction tasks, we do not make predictions at each time step in all cases. Hence, there is a need to create a concise representation for the entire sequence using the learned representations, for which we employ a dense interpolated embedding scheme, that encodes partial temporal ordering.

**Dense Interpolation for Encoding Order:** The simplest approach to obtain a unified representation for a sequence, while preserving order, is to simply concatenate embeddings at every time step. However, in our case, this can lead to a very high-dimensional, "cursed" representation which is not suitable for learning and inference. Consequently, we propose to utilize a dense interpolation algorithm from language modeling. Besides providing a concise representation, (Trask, Gilmore, and Russell 2015) demonstrated that the dense interpolated embeddings better encode word structures which are useful in detecting syntactic features. In our architecture, dense interpolation embeddings, along with the positional encoding module, are highly effective in capturing enough temporal structure required for even challenging clinical prediction tasks.

The pseudocode to perform dense interpolation for a given sequence is shown in Algorithm 2. Denoting the hidden representation at time $t$, from the attention

Dense Interpolation Embedding

**Input :** Steps $t$ of the time series and length of the sequence $T$, embeddings at step $t$ as $\mathbf{s}_t$, factor $M$.

**Output :** Dense interpolated vector representation $\mathbf{u}$.

**for** $t = 1$ *to* $T$ **do**
  $s = M * t/T$
  **for** $m = 1$ *to* $M$ **do**
    $w = pow(1 - abs(s - m)/M, 2)$ $\mathbf{u}_m = \mathbf{u}_m + w * \mathbf{s}_t$
  **end**
**end**

**Algorithm 2:** Dense interpolation embedding with partial order for a given sequence.

model, as $\mathbf{s}_t \in \mathbb{R}^d$, the interpolated embedding vector will have dimension $d \times M$, where $M$ is the *dense interpolation factor*. Note that when $M = T$, it reduces to the concatenation case. The main idea of this scheme is to determine weights $w$, denoting the contribution of $\mathbf{s}_t$ to the position $m$ of the final vector representation $\mathbf{u}$. As we iterate through the time-steps of a sequence, we obtain $s$, the relative position of time step $t$ in the final representation $\mathbf{u}$ and $w$ is computed as $w = (1 - \frac{|s-m|}{M})^2$. We visualize the dense interpolation process in Figure 17 for the toy case of $T = 5, M = 3$. The larger weights in $w$ are indicated by darker edges while the lighter edges indicates lesser influence. In practice, dense interpolation is implemented efficiently by caching $w$'s into a matrix $\mathbf{W} \in \mathbb{R}^{T \times M}$ and then performing the following matrix multiplication: $\mathbf{U} = \mathbf{S} \times \mathbf{W}$, where $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_T]$. Finally we can obtain $\mathbf{u}$ by stacking columns of $\mathbf{U}$.

**Linear and Softmax layers:** After obtaining a single vector representation from dense interpolation, we utilize a linear layer to obtain the logits. The final layer depends on the specific task. We can use a softmax layer for the binary classification problems, a sigmoid layer for multi-label classification since the classes are not mutually

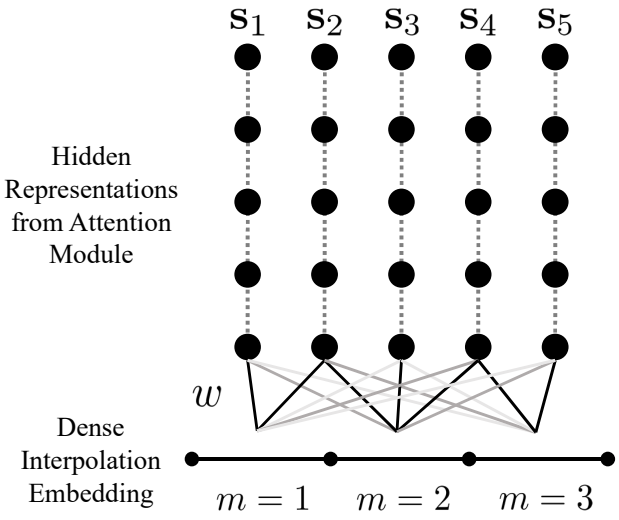



Figure 17: Visualizing the dense interpolation module, for the case when $T = 5$ and $M = 3$.

exclusive and a ReLU layer for regression problems. The corresponding loss functions are:

- *Binary classification*: $-(y \cdot \log(\hat{y})) + (1 - y) \cdot \log(1 - \hat{y})$, where $y$ and $\hat{y}$ are the true and predicted labels.
- *Multi-label classification*: $\frac{1}{K} \sum_{k=1}^{K} -(y_k \cdot \log(\hat{y}_k) + (1 - y_k) \cdot \log(1 - \hat{y}_k))$, where $K$ denotes the total number of labels in the dataset.
- *Regression*: $\sum_{t=1}^{T} (l_t - \hat{l}_t)^2$, where $l_t$ and $\hat{l}_t$ denote the true and predicted response variables at time-step $t$.

**Regularization:** In the proposed approach, we apply the following regularization strategies during training: (i) We apply dropout to the output of each sub-layer in the attention module prior to residual connections and normalize the outputs. We include an additional dropout layer after adding the positional encoding to the input embeddings, (ii) We also perform attention dropout, similar to (Vaswani et al. 2017), after computing the self-attention weights.

**Complexity:** Learning long-range dependencies is a key challenge in many sequence modeling tasks. Another notion of complexity is the amount of computation that can be parallelized, measured as the minimum number of sequential operations required. Recurrent models require $O(T)$ sequential operations with a total $O(T \cdot d^2)$ computations in each layer. In comparison, the proposed approach requires a constant $O(1)$ sequential operations (entirely parallelizable) with a total $O(T \cdot r \cdot d)$ computations per layer, where $r$ denotes the size of the mask for self-attention. In all our implementations, $d$ is fixed at 256 and $r \ll d$, and as a result our approach is significantly faster than RNN training.

### 6.1.2 MIMIC-III Benchmarks & Formulation

In this section, we describe the MIMIC-III benchmark tasks and the application of the *SAnD* framework to these tasks, along with a joint multi-task formulation.

The MIMIC-III database consists of de-identified information about patients admitted to critical care units between 2001 and 2012 (Johnson et al. 2016). It encompasses an array of data types such as diagnostic codes, survival rates, and more. Following (Harutyunyan et al. 2017), we used the cohort of $33,798$ unique patients with a total of $42,276$ hospital admissions and ICU stays. Using raw data from Physionet, each patient's data has been divided into separate episodes containing both time-series of events, and episode-level outcomes (Harutyunyan et al. 2017). The time-series measurements were then transformed into a 76-dimensional vector at each time-step. The size of the benchmark dataset for each task is highlighted in Table 7.

**In Hospital Mortality:** Mortality prediction is vital during rapid triage and risk/severity assessment. In Hospital Mortality is defined as the outcome of whether a patient dies during the period of hospital admission or lives to be discharged. This problem is posed as a binary classification one where each data sample spans a 24-hour

time window. True mortality labels were curated by comparing date of death (DOD) with hospital admission and discharge times. The mortality rate within the benchmark cohort is only 13%.

**Decompensation:** Another aspect that affects treatment planning is deterioration of organ functionality during hospitalization. Physiologic decompensation is formulated as a problem of predicting if a patient would die within the next 24 hours by continuously monitoring the patient within fixed time-windows. Therefore, the benchmark dataset for this task requires prediction at each time-step. True decompensation labels were curated based on occurrence of patient's DOD within the next 24 hours, and only about 4.2% of samples are positive in the benchmark.

**Length of Stay:** Forecasting length of a patient's stay is important in healthcare management. Such an estimation is carried out by analyzing events occurring within a fixed time-window, once every hour from the time of admission. As part of the benchmark, hourly remaining length of stay values are provided for every patient. These true range of values were then transformed into ten buckets to repose this into a classification task, namely: a bucket for less than a day, seven one day long buckets for each day of the 1st week, and two outlier buckets-one for stays more than a week but less than two weeks, and one for stays greater than two weeks (Harutyunyan et al. 2017).

**Phenotyping:** Given information about a patient's ICU stay, one can retrospectively predict the likely disease conditions. This process is referred to as acute care phenotyping. The benchmark dataset deals with 25 disease conditions of which 12 are critical such as respiratory/renal failure, 8 conditions are chronic such as diabetes, atherosclerosis, and 5 are 'mixed' conditions such as liver infections. Typically,

Table 7: Task-specific sample sizes of MIMIC-III dataset.

| Benchmark | Train | Validation | Test |
|---|---|---|---|
| Mortality | 14,659 | 3,244 | 3,236 |
| Decompensation | 2,396,001 | 512,413 | 523,208 |
| Length of Stay | 2,392,950 | 532,484 | 525,912 |
| Phenotyping | 29,152 | 6,469 | 6,281 |

a patient is diagnosed with multiple conditions and hence this can be posed as a multi-label classification problem.

### 6.1.3 Applying *SAnD* to MIMIC-III Tasks

In order to solve the afore-mentioned benchmark tasks with *SAnD*, we need to make a few key parameter choices for effective modeling. These include: size of the self-attention mask ($r$), dense interpolation factor ($M$) and the number of attention blocks ($N$). While attention models are computationally more efficient than RNNs, their memory requirements can be quite high when $N$ is significantly large. However, in practice, we are able to produce state-of-the-art results with small values of $N$. As described in the previous section, the total number of computations directly relies on the size of the mask, $r$ and interestingly our experiments show that smaller mask sizes are sufficient to capture all required dependencies in 3 out of 4 tasks, except phenotyping, which needed modeling of much longer-range dependencies. The dependency of performance on the dense interpolation factor, $M$ is more challenging to understand, since it relies directly on the amount of variability in the measurements across the sequence. The other hyperparameters of network such as the learning rate, batch size and embedding sizes were determined using the validation data. Note, in all cases, we used the Adam optimizer (Kingma and Ba 2014) with parameters $\beta_1 = 0.9, \beta_2 = 0.98$ and $\epsilon = 10^{-8}$. The training was particularly challenging for

the decompensation and length of stay tasks because of the large training sizes. Consequently, training was done by dividing the data into *chunks* of 20000 samples and convergence was observed with just 20-30 randomly chosen chunks. Furthermore, due to the imbalance in the label distribution, using a larger batch size (256) helped in some of the cases.

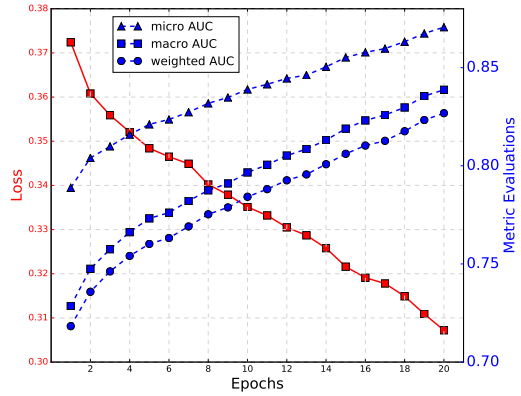
**Multi-task Learning:** In several recent results from the deep learning community, it has been observed that joint inferencing with multiple related tasks can lead to superior performance in each of the individual tasks, while drastically improving the training behavior. Hence, similar to the approach in (Harutyunyan et al. 2017), we implemented a multi-task version of our approach, *SAnD-Multi*, that uses a loss function that jointly evaluates the performance of all tasks, which can be expressed as follows:

$$\ell_{mt} = \lambda_p \ell_{ph} + \lambda_i \ell_{ihm} + \lambda_d \ell_{dc} + \lambda_l \ell_{los}, \tag{6.2}$$

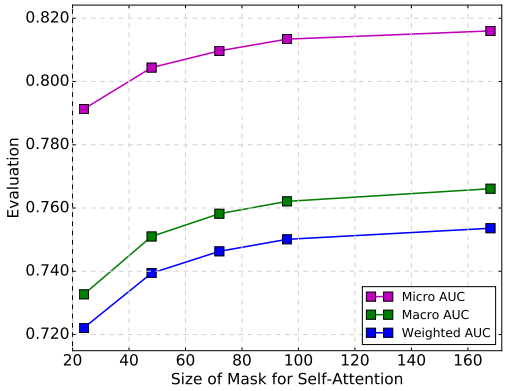
where $\ell_{ph}, \ell_{ihm}, \ell_{dc}, \ell_{los}$ correspond to the losses for the four tasks. The input embedding and attention modules are shared across the tasks, while the final representations and the prediction layers are unique to each task. Our approach allows the use of different mask sizes and interpolation factors for each task, but requires the use of the same $N$.

### 6.1.4 Performance Evaluation

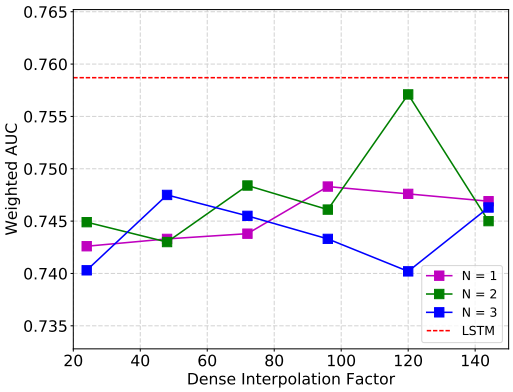
In this section we evaluate the proposed *SAnD* framework on the benchmark tasks and present comparisons to the state-of-the-art RNNs based on LSTM (Harutyunyan et al. 2017), and baseline logistic regression (LR) with hand-engineered features. To this end, we discuss the evaluation metrics and the choice of algorithm parameters. In particular, we analyze the impact of the choice of number of attention layers $N$, the dense interpolation factor $M$, and the mask size of the self-attention mechanism $r$ on



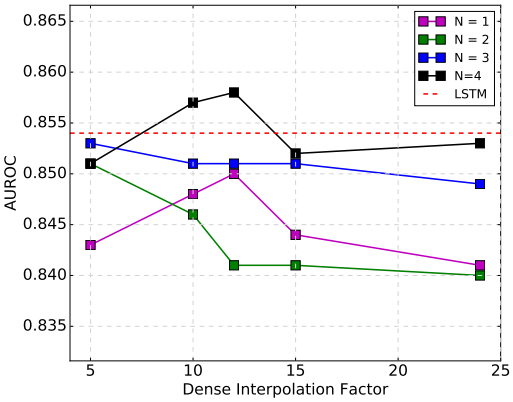

(a) *Phenotyping*: Convergence Behavior



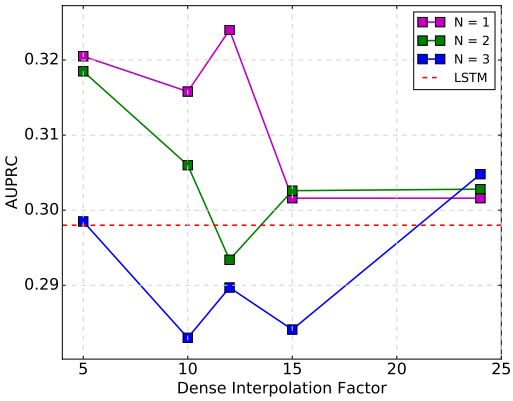

(b) *Phenotyping*: Choice of $r$



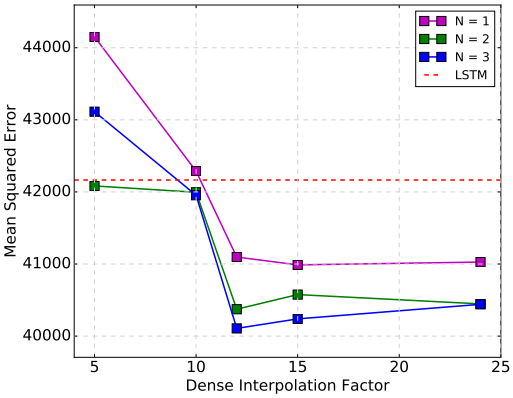

(c) *Phenotyping*: Choice of $(N, M)$




(d) *In Hospital Mortality*: Choice of $(N, M)$




(e) *Decompensation*: Choice of $(N, M)$




(f) *Length of Stay*: Choice of $(N, M)$

Figure 18: Applying *SAnD* to MIMIC-III benchmark tasks - We illustrate the training behavior and impact of the choice of the attention mask size, number of attention layers and dense interpolation factor on test performance.

Table 8: Performance Comparison for the MIMIC-III benchmark tasks, using both single-task and multi-task strategies.

| Metrics | Method | | | | |
|---|---|---|---|---|---|
| | **LR** | **LSTM** | ***SAnD*** | **LSTM-Multi** | ***SAnD*-Multi** |
| **Task 1: Phenotyping** | | | | | |
| Micro AUC | 0.801 | **0.821** | 0.816 | 0.817 | 0.819 |
| Macro AUC | 0.741 | **0.77** | 0.766 | 0.766 | **0.771** |
| Weighted AUC | 0.732 | 0.757 | 0.754 | 0.753 | **0.759** |
| **Task 2: In Hospital Mortality** | | | | | |
| AUROC | 0.845 | 0.854 | 0.857 | **0.863** | 0.859 |
| AUPRC | 0.472 | 0.516 | 0.518 | 0.517 | **0.519** |
| min(Se, P+) | 0.469 | 0.491 | 0.5 | 0.499 | **0.504** |
| **Task 3: Decompensation** | | | | | |
| AUROC | 0.87 | 0.895 | 0.895 | 0.900 | **0.908** |
| AUPRC | 0.2132 | 0.298 | 0.316 | 0.319 | **0.327** |
| min(Se, P+) | 0.269 | 0.344 | 0.354 | 0.348 | **0.358** |
| **Task 4: Length of Stay** | | | | | |
| Kappa | 0.402 | 0.427 | **0.429** | 0.426 | **0.429** |
| MSE | 63385 | 42165 | 40373 | 42131 | **39918** |
| MAPE | 573.5 | 235.9 | 167.3 | 188.5 | **157.8** |

the test performance. Finally, we report the performance of the mutli-task variants of both RNN and proposed approaches on all tasks.

#### 6.1.4.1 Single-Task Case

**Phenotyping:** This multi-label classification problem involves retrospectively predicting acute disease conditions. Following (Lipton et al. 2015) and (Harutyunyan et al. 2017), we use the following metrics to evaluate the different approaches on this task: (i) macro-averaged Area Under the ROC Curve (AUROC), which averages per-label AUROC, (ii) micro-averaged AUROC, which computes single AUROC score

for all classes together, (iii) weighted AUROC, which takes disease prevalence into account. The learning rate was set to 0.0005, batch size was fixed at 128 and a residue dropout probability of 0.4 was used. First, we observe that the proposed attention model based architecture demonstrates good convergence characteristics as shown in Figure 18a. Given the uneven distribution of the class labels, it tends to overfit to the training data. However, with both attention and residue dropout regularizations, it generalizes well to the validation and test sets. Since, the complexity of the proposed approach relies directly on the attention mask size ($r$), we studied the impact of $r$ on test performance. As shown in Figure 18b, this task requires long-term dependencies in order to make accurate predictions. Though all performance metrics improve upon the increase of $r$, there is no significant improvement beyond $r = 96$ which is still lower than the feature dimensionality 256. As shown in Figure 18c, using a grid search on the parameters $N$ (number of attention layers) and $M$ (dense interpolation factor), we identified the optimal values. As described earlier, lowering the value of $N$ reduces the memory requirements of *SAnD*. In this task, we observe that the values $N = 2$ and $M = 120$ produced the best performance, and as shown in Table 8, it is highly competitive to the state-of-the-art results.

**In Hospital Mortality:** In this binary classification task, we used the following metrics for evaluation: (i) Area under Receiver Operator Curve (AUROC), (ii) Area under Precision-Recall Curve (AUPRC), and (iii) minimum of precision and sensitivity (Min(Se,P+)). In this case, we set the batch size to 256, residue dropout to 0.3 and the learning rate at 0.0005. Since the prediction is carried out using measurements from the last 24 hours, we did not apply any additional masking in the attention module, except for ensuring causality. From Figure 18d, we observe that the best performance was obtained at $N = 4$ and $M = 12$. In addition, even for the optimal

$N$ the performance drops with further increase in $M$, indicating signs of overfitting. From Table 8, it is apparent that *SAnD* outperforms both the baseline methods.

**Decompensation:** Evaluation metrics for this task are the same as the previous case of binary classification. Though we are interested in making predictions at every time step of the sequence, we obtained highly effective models with $r = 24$ and as a result our architecture is significantly more efficient for training on this large-scale data when compared to an LSTM model. Our best results were obtained from training merely on about 25 chunks (batch size = 128, learning rate = 0.001) , when $N = 1$ and $M = 10$ (see Figure 18e), indicating that increasing the capacity of the model easily leads to overfitting. This can be attributed to the heavy bias in the training set towards the negative class. Results for this task (Table 8) are significantly better than the state-of-the-art, thus evidencing the effectiveness of *SAnD*.

**Length of Stay:** Since this problem is solved as a multi-class classification task, we measure the inter-agreement between true and predicted labels using the Cohen's linear weighted kappa metric. Further, we assign the mean length of stay from each bin to the samples assigned to that class, and use conventional metrics such as mean squared error (MSE) and mean absolute percentage error (MAPE). The grid search on the parameters revealed that the best results were obtained at $N = 3$ and $M = 12$, with no further improvements with larger $N$ (Figure 18f). Similar to the decompensation case, superior results were obtained using $r = 24$ when compared with the LSTM performance, in terms of all the evaluation metrics.

#### 6.1.4.2 Multi-Task Case

We finally evaluate the performance of *SAnD*-Multi by jointly inferring the model parameters with the multi-task loss function in Eq (6.2). We used the weights $\lambda_p = 0.8, \lambda_i = 0.5, \lambda_d = 1.1, \lambda_l = 0.8$. Interestingly, in the multi-task case, the best

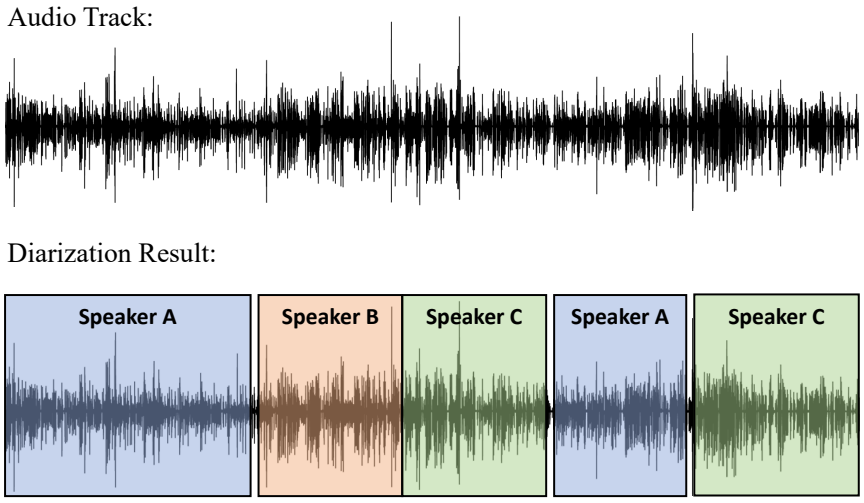



Figure 19: Illustration of the speaker diarization problem.

results for phenotyping were obtained with a much lower mask size (72), thereby making the training more efficient. The set of hyperparameters were set at batch size = 128, learning rate = 0.0001, $N = 2$, $M = 36$ for phenotyping and $M = 12$ for the other three cases. As shown in Table 8, this approach produces the best performance in almost all cases, with respect to all the evaluation metrics.

## 6.2 Speaker diarization using triplet network

With the ever-increasing volume of multimedia content on the Internet, there is a crucial need for tools that can automatically index and organize the content. In particular, speaker diarization deals with the problem of indexing speakers in a collection of recordings, without *a priori* knowledge about the speaker identities. As illustrated in Figure 19, in scenarios where the single-speaker assumption of recognition systems is violated, it is critical to first separate speech segments from different speakers prior to downstream processing. Note that diarization is treated as an unsupervised learning problem, in contrary to speaker identification, which requires training on extensive labeled data, and speaker verification, which solely returns yes/no for a

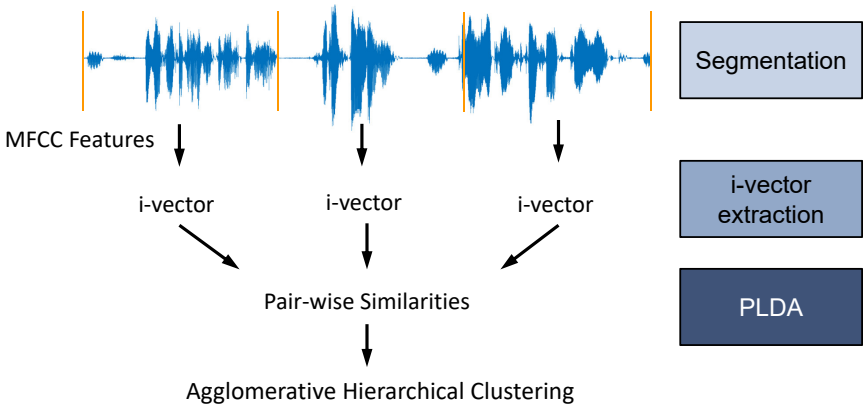



Figure 20: Conventional diarization approach based on i-vectors.

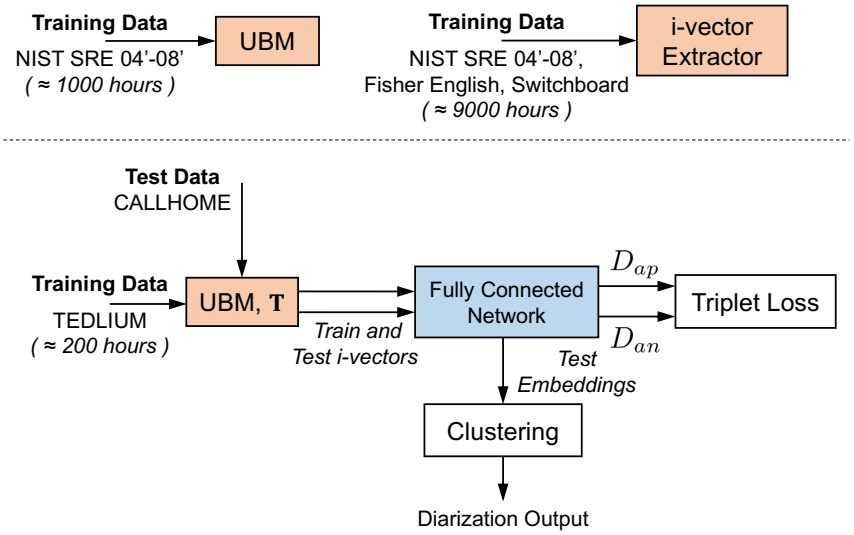
certain speaker (Reynolds 1995). Typical challenges in speaker diarization include the need to deal with similarities between a large set of speakers, differences in acoustic conditions, and the adaptation of a trained system to new speaker sets.

An important class of diarization approaches rely on extracting i-vectors to represent speech segments, and then scoring similarities between i-vectors using pre-defined similarity metrics (e.g. cosine distance) to achieve speaker discrimination. Despite its widespread use, it is well known that the i-vector extraction process requires extensive training of a Gaussian Mixture Model based Universal Background Model (GMM-UBM) and estimation of the total variability matrix (i-vector extractor) beforehand using large corpora of speech recordings. While several choices for the similarity metric currently exist, likelihood ratios obtained through a separately trained Probabilistic Linear Discriminant Analysis (PLDA) model are commonly utilized (Khoury et al. 2014). The process of such i-vector based approaches are demonstrated in Figure 20.

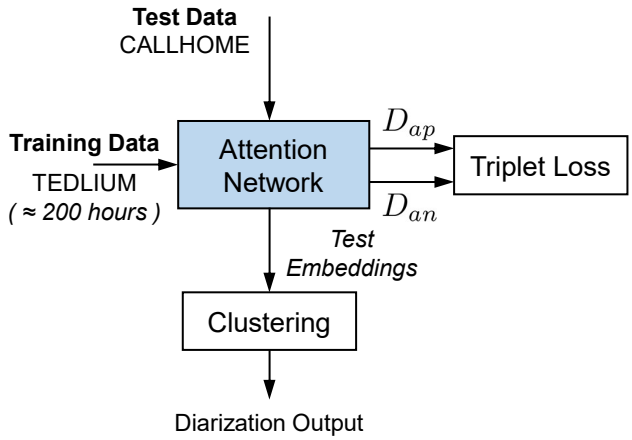
More recently, with the advent of modern representation learning paradigms, designing effective metrics for comparing i-vectors has become an active research direction. In particular, inspired by its success in computer vision tasks (Schroff, Kalenichenko, and Philbin 2015; Lin et al. 2015; Hoffer and Ailon 2015), many recent

efforts formulate the diarization problem as deep metric learning (Le Lan et al. 2017; Garcia-Romero et al. 2017; Bredin 2017b). For instance, a triplet network that builds latent spaces, wherein a simple Euclidean distance metric is highly effective at separating different classes, is a widely adopted architecture. However, in contrast to its application in vision tasks, metric learning is carried out on the i-vector representations instead of the raw data (Le Lan et al. 2017). Consequently, the first stage of the diarization pipeline stays intact, while the second stage is restricted to using fully connected networks. Though this modification produced state-of-the-art results in diarization and outperformed conventional scoring strategies, it does not support joint representation and task-based learning, which has become the *modus operandi* in deep learning. On the other hand, Garcia-Romero et al. (Garcia-Romero et al. 2017) propose to perform joint embedding and metric learning, but use siamese networks for metric learning, which have generally shown poorer performance when compared to triplet networks (Hoffer and Ailon 2015).

In this section, we propose to explore the use of joint representation learning and similarity metric learning with triplet loss in speaker diarization, while entirely dispensing the need for i-vector extraction. Encouraged by the recent success of *self-attention* mechanism in sequence modeling tasks (Vaswani et al. 2017; Song, Rajan, et al. 2018), for the first time, we leverage attention networks to model the temporal characteristics of speech segments. Experimental results on the CALLHOME corpus demonstrate that, with an appropriate embedding architecture, triplet network applied on raw audio features from a comparatively smaller dataset outperforms the same applied on i-vectors, wherein the GMM-UBM was trained using a much larger corpus.




(a) Baseline approach: (top) i-vectors are extracted using a large corpus of recordings with GMM-UBM and i-vector extractor modules; (bottom) Similarity metric is trained using a triple network trained on the i-vectors.




(b) Proposed approach: Joint learning of embedding and similarity metric for diarization. As observed, it completely eliminates the i-vector extraction process and enables effective training with limited data.

Figure 21: Comparison of diarization strategies and training data requirements for the baseline approach in (Le Lan et al. 2017) and the proposed approach.

### 6.2.1 Proposed Approach

As shown in Figure 26(b), the proposed approach works directly with raw temporal speech features to learn a similarity metric for diarization. Compared to the baseline in Figure 26(a), the two-stage training process is simplified into a single end-to-end learning strategy, wherein deep attention models are used for embedding computation and the triplet loss is used to infer the metric. Similar to existing diarization paradigms, we first train our network using out-of-domain labeled corpus, and then perform diarization on a target dataset using unsupervised clustering. In the rest of this section, we describe the proposed approach in detail.

#### 6.2.1.1 Temporal Segmentation and Feature Extraction

For the speech recordings, we first perform non-overlapping temporal segmentation into 2-second segments. Following the Voice Biometry Standardization (VBS) [4], we extract MFCC features using 25ms Hamming windows with 15ms overlap. After adding delta and double-delta coefficients, we obtain 60-dimensional feature vectors at every frame. Consequently, each data sample corresponds to a temporal sequence feature $\mathbf{x}_i \in \mathbb{R}^{T \times d}$, where $T$ is the number of frames in each segment and $d = 60$ is the feature dimension.

#### 6.2.1.2 Embeddings using Attention Models

As described earlier, we use attention models to learn embeddings directly from MFCC features for the subsequent metric learning task. The attention model used in our architecture is illustrated in Figure 22. The module comprised of a multi-head, self-attention mechanism is the core component of the attention model (Vaswani et al.

[4] http://voicebiometry.org/

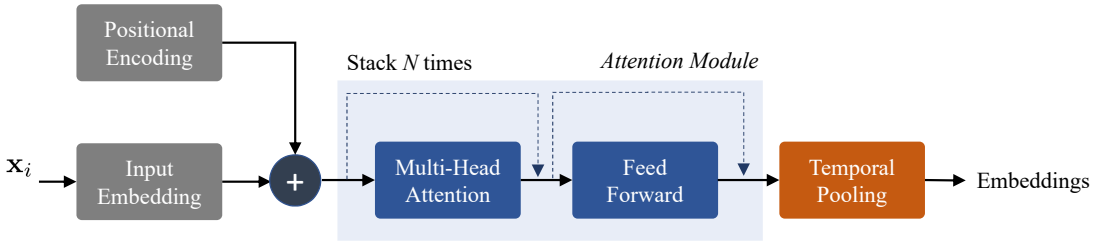



Figure 22: Illustration of the attention model used for computing embeddings from MFCC features of speech segments.

2017). More specifically, denoting the input representation at layer $\ell$ as $\{\mathbf{h}_t^{\ell-1}\}_{t=1}^T$, we can obtain the hidden representation at time step $i$ based on attention as follows:

$$\mathbf{h}_i^\ell = \sum_{t=1}^{T} w_t^{(i)} \mathbf{h}_t^{\ell-1}, \quad 1 \leq i \leq T, \tag{6.3}$$

$$w_t^{(i)} = \text{softmax}\left(\frac{\mathbf{h}_i^{\ell-1} \cdot \mathbf{h}_t^{\ell-1}}{\sqrt{D}}\right), \tag{6.4}$$

$$\mathbf{h}_i^\ell = \mathcal{F}(\mathbf{h}_i^\ell), \tag{6.5}$$

Here, $D = 256$ refers to the size of the hidden layer and $\mathcal{F}$ denotes a feed-forward neural network. The attention weight in equation (6.4) denotes the interaction between temporal positions $i$ and $t$. During the computation of hidden representation at time step $i$, $w_t^{(i)}$ weights the contribution from other temporal positions. Note that, these representations are processed by $\mathcal{F}$ before connecting to the next attention module, as shown in Figure 22. We employ a 1D convolutional layer (kernel size is 1) with ReLU activation (Nair and Hinton 2010) for $\mathcal{F}$. Finally, the attention module is stacked $L$ times to learn increasingly deeper representations.

Attention-based representations in equation (6.3) are computed within each speech segment independently and hence this process is referred to as *self-attention*. Furthermore, the hidden representations $\mathbf{h}_t^\ell$ are computed using $H$ different network parameterizations, denoted as heads, and the resulting $H$ attention representations are concatenated together. This can be loosely interpreted as an ensemble of repre-

sentations. Such a *multi-head* operation facilitates dramatically different temporal parameterizations and significantly expands the modeling power. Our current implementation sets $L = 2$ and $H = 8$.

Although attention computation explicitly models the temporal interactions, it does not encode the crucial ordering information contained in speech. The front-end positional encoding block handles this problem by mapping every relative frame position $t$ in the segment to fixed locations in a random lookup table. As shown in Figure 22, the encoded representation is subsequently added up with the input embedding (obtained also from a 1D CNN layer). Finally, we include a temporal pooling layer to reduce the final representation $\mathbf{h}^L \in \mathbb{R}^{T \times D}$ into a $D$-dimensional vector by averaging along the time-axis.

#### 6.2.1.3 Metric Learning with Triplet Loss

The representations from the deep attention model are then used to learn a similarity metric with the triplet ranking loss. Note that the attention model parameters and the metric learner are optimized jointly using back-propagation. In a triplet network, each input is constructed as a set of 3 samples $\mathbf{x} = \{\mathbf{x}_p, \mathbf{x}_r, \mathbf{x}_n\}$, where $\mathbf{x}_r$ denotes an anchor, $\mathbf{x}_p$ denotes a positive sample belonging to the same class as $\mathbf{x}_r$ and $\mathbf{x}_n$ a negative sample from a different class. Each of the samples in $\mathbf{x}$ are processed using the attention model (Section 3.2) $\mathcal{A}(\cdot) : \mathbb{R}^{T \times d} \mapsto \mathbb{R}^D$ and distances are computed in the resulting latent spaces:

$$D_{rp} = \|\mathcal{A}(\mathbf{x}_r) - \mathcal{A}(\mathbf{x}_p)\|_2$$

$$D_{rn} = \|\mathcal{A}(\mathbf{x}_r) - \mathcal{A}(\mathbf{x}_n)\|_2$$

The triplet loss is defined as

$$l(\mathbf{x}_p, \mathbf{x}_r, \mathbf{x}_n) = \max(0, D_{rp}^2 - D_{rn}^2 + \alpha) \tag{6.6}$$

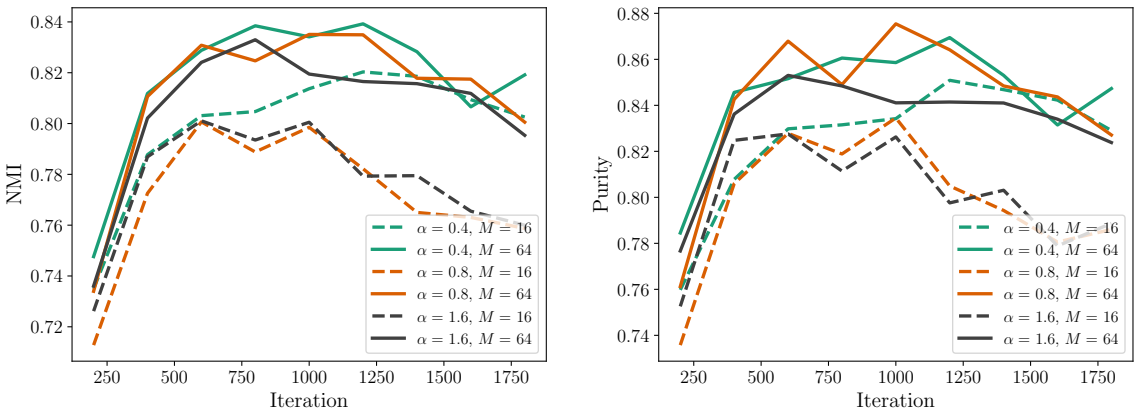
where $\alpha$ is the margin and the objective is to achieve $D_{rn}^2 \geq D_{rp}^2 + \alpha$. In comparison, the contrastive loss used in (Garcia-Romero et al. 2017) includes the hinge term $\max(0, \alpha - D_{ij})$ for different-class samples $\mathbf{x}_i$ and $\mathbf{x}_j$, and hence requires $\alpha$ to be a global margin. Such a formulation significantly restricts the model flexibility and expressive power.

Given a large number of samples $N$, the computation of equation (6.6) is infeasible among the $\mathcal{O}(N^3)$ triplet space. It is tempting to greedily select the most effective triplets, which maximizes $D_{rp}$ and minimizes $D_{rn}$. Instead of performing such hard sampling, we follow (Schroff, Kalenichenko, and Philbin 2015) to sample all possible $\mathbf{x}_p$ and only selecting *semi-hard* $\mathbf{x}_n$: the negative samples satisfying $D_{rp}^2 \leq D_{rn}^2 \leq D_{rp}^2 + \alpha$. Additionally, we adopt an online sampling strategy that restricts the sampling space to the current mini-batch during training. All sampled triplets are gathered to compute the loss in equation (6.6).

For the online sampling scheme, the mini-batch construction step is crucial. Ideally, each batch should cover both a large number of speakers and sufficient samples per speaker. However, we are constrained by the GPU memory (8GB) and only able to set maximum batch size $B = 256$. We preset $M$ as the number of speakers per batch and when sampling each mini-batch, $M$ speakers are first sampled and $B/M$ speech segments are then sampled for every speaker. As a result, the parameter $M$ represents the trade-off between modeling more speakers each time, and covering sufficient samples for those speakers. In our experiments, $M$ was tuned based on the performance on the development set, as will be discussed in Section 6.2.2.1.

### 6.2.2 Experiments

In this section, we discuss the training process for our approach and evaluate its performance on the CALLHOME corpus.




(a) NMI score from the speaker clustering results.

(b) Purity score from the speaker clustering results.

Figure 23: Parameter tuning on TEDLIUM development set for triplet margin $\alpha$ and number of speakers per batch $M$. Curves for $M = 8$ and 32 are omitted for clarity.

#### 6.2.2.1 Triplet Network Training

The proposed model was trained on the TEDLIUM corpus which consists of 1495 audio recordings. After ignoring speakers with less than 45 transcribed segments, we have a set of 1211 speakers with an average recording length of 10.2 minutes. All recordings were down-sampled to 8kHz to match the target CALLHOME corpus. The temporal segmentation and MFCC extraction were carried out as discussed in Section 6.2.1.1.

For the proposed approach, there are two important training parameters that need to be selected, i.e. triplet margin $\alpha$ and the number of speakers per mini-batch $M$. In order to quickly configure the parameters, we build a training subset by randomly selecting 20% of the total recordings and a development set by taking 50 recordings from the original TEDLIUM train, dev and test sets. At every 200 iterations of training on the subset, we extract the embeddings for the development set and perform speaker clustering using $k$-Means, with a known number of speakers. The

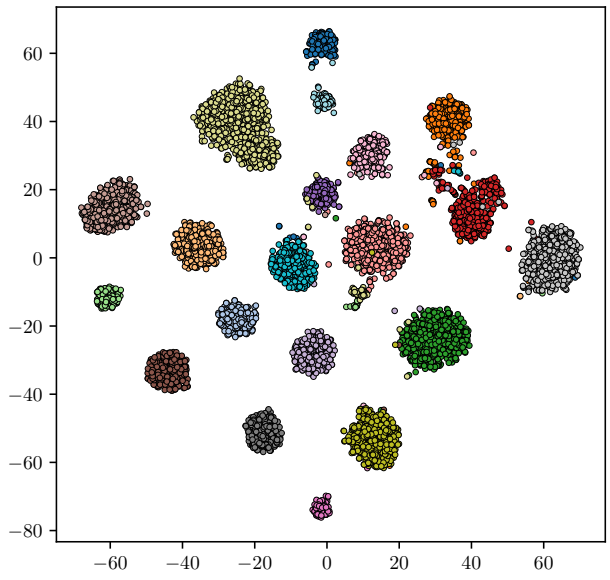



Figure 24: 2D t-SNE visualization of the first 20 speakers from TEDLIUM development set. Each point corresponds to one speech segment and they are color coded by the speaker.

clustering performance is evaluated by the standard Normalized Mutual Information (NMI) and Purity scores. Based on this procedure, we jointly tuned both parameters by performing a grid search on $\alpha = [0.4, 0.8, 1.6]$ and $M = [8, 16, 32, 64]$. As shown in Figure 23, having a higher $M$ value consistently provides better clustering results and alleviates model overfitting. Additionally, a lower triplet margin generally helps the training process. Based on these observations, we configured $\alpha = 0.8, M = 64$ to train our model on the entire TEDLIUM corpus.

To study the embeddings from the attention model and the impact of triplet loss, we show the 2D t-SNE visualization (Van Der Maaten 2014) of samples in the development set in Figure 24. It is observed that the model is highly effective at separating unseen speakers and provides little distinction on segments from the same speakers. These embeddings achieve 0.94 score on both NMI and Purity, with $k$-Means clustering for the development set.

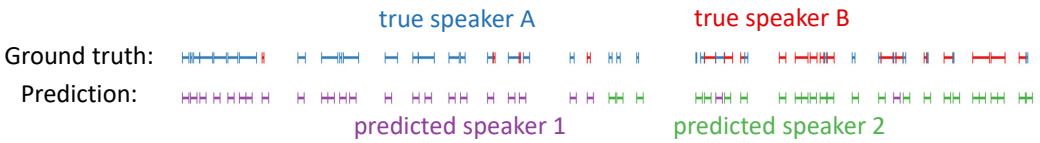



Figure 25: Diarization result on an example speech recording.

#### 6.2.2.2 Diarization Results

The trained model is evaluated on the CALLHOME corpus for diarization performance. CALLHOME consists of telephone conversations in 6 languages: Arabic, Chinese, English, German, Japanese and Spanish. In total, there are 780 transcribed conversations containing 2 to 7 speakers. After obtaining the embeddings through the proposed approach, we perform x-means (Pelleg and Moore 2000) to estimate the number of speakers and then use $k$-means clustering with the estimation. We force x-means to split at least 2 clusters by initializing it with 2 centroids. Note that there are usually multiple moving parts on complete diarization systems in the literature. In particular, more sophisticated clustering algorithms (Q. Wang et al. 2017), overlapping test segments and calibration (Sell and Garcia-Romero 2014) can be incorporated to improve the overall diarization performance. However, in this work we focus on investigating the efficacy of the DNN modeling and fix the other components in their basic configurations.

We first visually demonstrate the diarization result obtained by our proposed algorithm. As shown in Figure 25, the algorithm is able to successfully pick up the transition of speaker around the middle of the recording, while confusing the speaker identify only on a few speech segments. In order to quantitatively compare the diarization performance, we utilize the metric of Diarization Error Rate (DER) calculated by `pyannote.metric` (Bredin 2017a). Although DER collectively considers

Table 9: Diarization Results on CALLHOME Corpus.

| System | | DER (%) |
|---|---|---|
| i-vector | cosine | 18.7 |
| | PLDA (Khoury et al. 2014) | 17.6 |
| | Triplet with FCN (Le Lan et al. 2017) | 13.4 |
| **Proposed Approach** | | **12.7** |

false alarms, missed detections and confusion errors, most existing systems evaluated on CALLHOME (Sell and Garcia-Romero 2014; Garcia-Romero et al. 2017) accounts for only the confusion rate and ignores overlapping segments. Following this convention, we use the oracle speech activity regions and use only the non-overlapping sections. Additionally, there is a collar tolerance of 250ms at both beginning and end of each segment. We compare the proposed approach with the following baseline systems:

**Baseline 1: i-vector + cosine/PLDA scoring**. We utilize VBS pre-trained models for i-vector extraction on CALLHOME corpus. The specific GMM-UBM and i-vector extractor training data are shown in Figure 26(a). Though different from ours, the training corpus is significantly more comprehensive than the TEDLIUM set we used. The GMM-UBM consists of 2048 Gaussian components and the i-vectors are $600-$dimensional. We also used the backend LDA model contained in VBS for i-vector pre-processing. In the actual clustering, cosine or PLDA scores are used to calculate the sample-to-centroid similarities at each iteration.

**Baseline 2: i-vector + triplet with FCN training**. This baseline is very similar to (Le Lan et al. 2017) except for 2 modifications: 1) We do not consider the speaker linking procedure as there are very few repeated speakers in CALLHOME. 2) We use a larger FCN network than (Le Lan et al. 2017) to allow a fair comparison to the proposed approach. The hidden layers have size $512 - 1024 - 512 - 256$ and

batch normalization (Ioffe and Szegedy 2015) is applied at each layer after the ReLU activation. Further, i-vectors are extracted on TEDLIUM based on the transcribed speech sections with average length of 8.6 seconds. The triplet network is tuned in a similar procedure as in Section 6.2.2.1 and the best parameters were found to be $\alpha = 0.4, M = 16$.

The comparison between the proposed approach and the baselines is shown in Table 9. It is observed that baseline 2 indeed exceeds both conventional i-vector scoring methods. However, our unified learning approach trained on a much smaller TEDLIUM corpus achieves better performance, this evidencing the effectiveness of end-to-end learning.

Chapter 7

# NODE EMBEDDING ON MULTI-LAYER GRAPHS FOR COMMUNITY DETECTION

When the graph structure exists in multiple data sources, it is beneficial to consider the fusion strategy under the multilayer graph setting. In this chapter, we focus on the crucial problem of community detection in graph mining. Inside the multilayer graph structure, each layer characterizes specific kind of relationship and the whole graph is a comprehensive representation of the overall relationships. Inspired by the recent advances which adopted neural word embedding to graphs for representation learning (see Section 2.5), we propose a systematic fusion approach to extend graph embedding to the case of multilayer graphs for improved community detection.

## 7.1 Background on Neural Embedding for Graph Vertices

In this section, we review the DeepWalk algorithm proposed by Perozzi *et al.* in (Perozzi, Al-Rfou, and Skiena 2014) for obtaining dense embeddings of nodes in a graph, which can be subsequently used for inferencing tasks such as clustering.

The DeepWalk algorithm utilizes neural optimization techniques originally designed for language modeling to obtain dense, low-dimensional embeddings for nodes in a graph. In language modeling, the goal is to learn a low-dimensional vector representation for natural word, based on the context, from a large corpus. The latent representations are known to reveal rich semantic information and can estimate the likelihood of specific sequence of words appearing in the corpus. Similar to word embeddings in the NLP literature, the distances latent dimensions provide a convenient metric for understanding similarities between nodes in the network. Furthermore,

such continuous vector spaces enable the definition of smooth decision boundaries between different communities and groups with homogeneous behavior. Given a graph $G$ and the binary adjacency matrix of size $|\mathcal{V}| \times |\mathcal{V}|$, the goal is to generate latent representations, $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $d$ is the number of dimensions of the embedding and $|\mathcal{V}|$ indicates the cardinality of the vertex set.

The central idea of the DeepWalk algorithm is to use short streams of randomly-generated walks to define the notion of context for each of the vertices. Random walk based methods have had long-standing success in quantifying similarities between entities in graph structured data, for example community detection. Formally, a random walk is a stochastic process with a set of random variables defined as vertices chosen at random from the neighbors of each vertex in the sequence. The ability of random walk to reveal the local structure makes it a natural tool for extracting information from graphs. Let us consider a simple metric walk $\mathcal{W}_t$ in step $t$, which is rooted at the vertex $v_i$. The transition probability between the nodes $v_i$ and $v_j$ can be expressed as

$$P(\mathcal{W}_t = v_j | \mathcal{W}_{t-1} = v_i) = h(\|\mathbf{x}_i - \mathbf{x}_j\|_2 / \sigma), \tag{7.1}$$

where $\|\mathbf{x}_i - \mathbf{x}_j\|_2$ indicates the similarity metric between the two vertices in the latent space to be recovered and $h$ is a linking function that connects the vertex similarity to the actual co-occurrence probability. Interestingly, with appropriate choice of length of the walks, the true metric can be recovered accurately from the co-occurrence statistics constructed using random walks. Furthermore, in (Perozzi, Al-Rfou, and Skiena 2014), the authors note that the frequency which vertices appear in the short random walks follows a power-law distribution, similar to words in natural language corpora. This naturally motivates the use of ideas from neural language modeling.

Given a sequence of words of length $k$, $(w_0, w_1, \cdots, w_k)$, where $w_i$ denotes a word

in the vocabulary, neural word embeddings attempt to obtain vector spaces that can recover the likelihood of observing a word given its context, i.e., $P(w_k|w_0, w_1, \cdots, w_{k-1})$ over all sequences in the training corpus. Extending this idea to the case of graphs, a random walk on the nodes, starting from the node $v_i$ produces the sequence $\mathcal{W}$, analogous to sentences in language data. In particular, a simple uniform sampling method can be employed to select the next vertex in the sequence, based on the neighbors of the current vertex. They start with a node at random say $v_i$, which is the current node, then randomly select another node $v_j$ from the neighbors of $v_i$. Now, $v_j$ is the current node and this process is repeated till a desired length is reached. Alternately, more sophisticated weighting strategies based on graph traversal can also be used (Grover and Leskovec 2016) $v_k$ given the set of previously visited nodes, $(v_1, v_2, \cdots, v_{k-1})$.

To briefly summarize the the DeepWalk algorithm, it begins by generating a large stream of random walk for a given $G$ based on uniform random sampling at each vertex. Given all the sequences, DeepWalk treats them as meaningful instances from a language and directly employs skipgram or the hierarchical softmax language models to obtain dense embeddings for the nodes.

## 7.2 Proposed approach

In this section, we start by presenting an overview of the proposed approach for generating unified, latent representations from multilayer graphs. Then we will describe in detail the early fusion and late fusion stages. Assume that we have access to $M$ different attributes for defining the edge sets between a set of nodes $\mathcal{V}$. While naive approaches such as the integration of the edge affinities, or concatenation of latent representations for each of the nodes can be used to merge the information

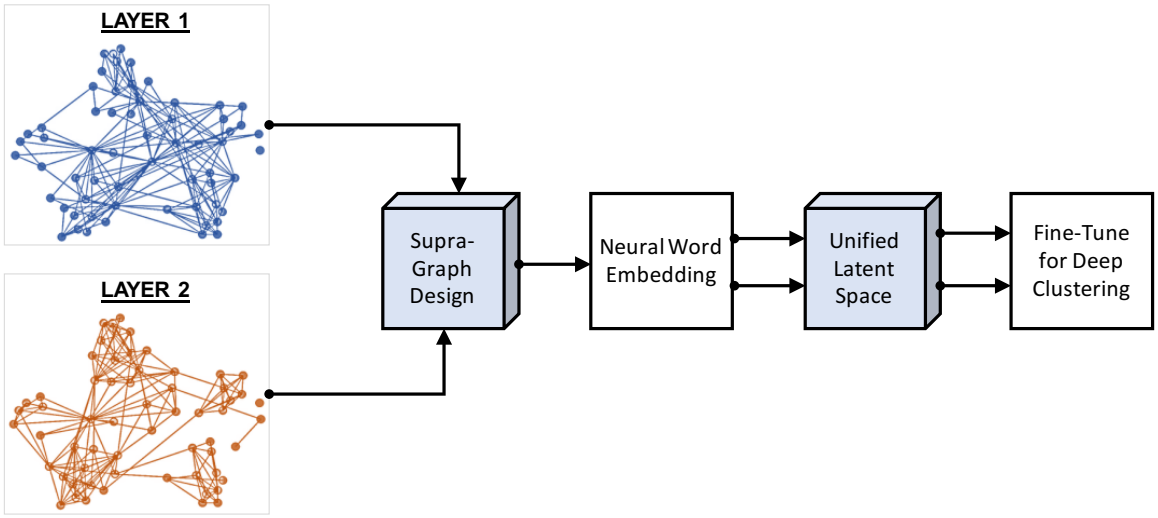



Figure 26: Proposed approach for extracting deep embeddings from multi-layer graphs. While the early fusion stage constructs a supra adjacency matrix by computing transition probabilities across the different layers, the late fusion stage employs a deep embedding framework that fuses the latent representations from the different layers and optimizes the network parameters using a deep clustering objective.

from different layers, we argue that a systematic approach for extracting deep, unified embeddings can lead to significant improvements in the subsequent inferencing tasks.

As illustrated in Figure 26, our approach takes as input the set of graphs $(G^{(m)}, \mathcal{V}, \mathcal{E}^{(m)}), \forall m = 1, \cdots, M$ and provides $d$-dimensional latent representations for each of the nodes in $\mathcal{V}$. If the input graphs contain weighted edges, one common practice is to employ a simple thresholding scheme to binarize the edge weights. Note that, when the DeepWalk algorithm is applied on real-world graphs after thresholding, the information about the varying density of edges in different regions of the data domain are lost, thereby leading to sub-optimal performances.

The first stage of our approach performs early fusion prior to obtaining the latent representations. More specifically, we merge the information across layers by constructing a supra adjacency matrix encoding multilayer transitions, similar to the approach in (Kuncheva and Montana 2015). In this process, the nodes with

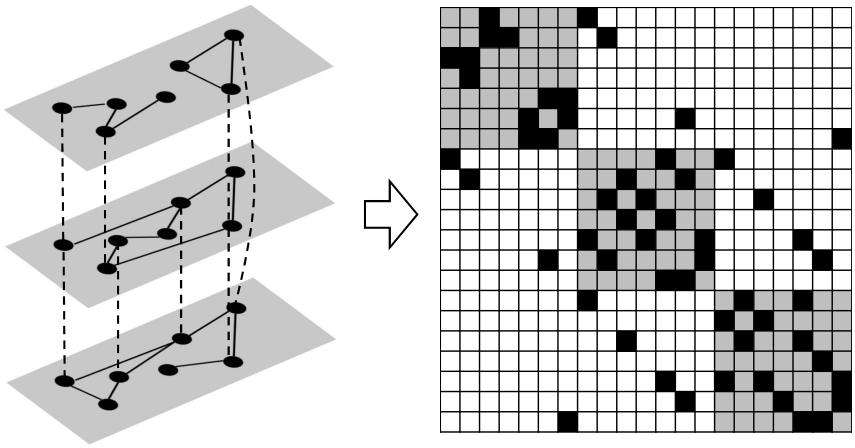
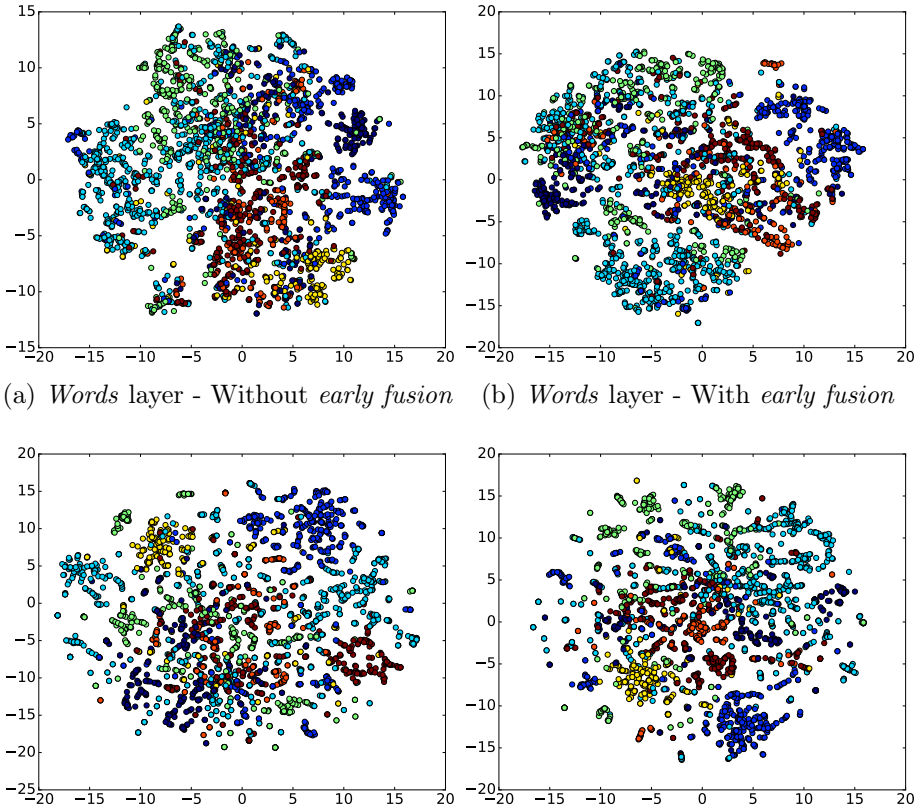


Figure 27: *Early fusion*: We introduce edges between a node and its counterparts in different layers to exploit the shared local structure for inferring robust latent representations. Here, we illustrate the supra graph construction process for an example case with 7 nodes and 3 layers. While the block diagonals correspond to the adjacency matrices of the individual layers, the off-diagonal entries encode the inter-layer edges.

similar local community structured are tied across different layers. Following this, the DeepWalk algorithm is employed directly on the supra graph, which produces $M$ different embeddings for each of the nodes in $\mathcal{V}$. While concatenating the latent representations from different layers is the widely adopted strategy in multi-modal fusion tasks, we show that a principled late fusion strategy can produce more effective unified representations for the nodes. In particular, we develop a deep embedding architecture based on a recently proposed clustering objective (Xie, Girshick, and Farhadi 2016), and utilize a multi-input multi-output autoenoder to initialize the parameters of the network. From our experimental results, we observe that the proposed two-stage approach produces highly robust latent representations and hence improved clustering performance.



(a) *Words* layer - Without *early fusion* (b) *Words* layer - With *early fusion*

(c) *Citations* layer - Without *early fusion* (d) *Citations* layer - With *early fusion*

Figure 28: Visualization of the latent representations obtained for the nodes in the Cora citation dataset, using the words and citation attributes respectively. The 2-D embeddings for the latent features were created using the t-SNE algorithm and the nodes are colored by their true cluster label. Using the early fusion strategy exploits the local community structure and leads to tighter grouping of communities (e.g. blue cluster in (b) and cyan cluster in (d)).

### 7.2.1 Early Fusion - DeepWalk with Supra Graphs

The early fusion step aims at modifying the graph structure to both exploit the community structure shared across different layers and benefit the subsequent late fusion stage. Since the quality of the random walks are crucial to the deep feature extraction process, and the walks are entirely determined by the graph structure, this step is critical in the overall approach. For example, let us consider the problem

of detecting communities from a network. When nodes in a multilayer graph form communities, there can be two scenarios: (a) the nodes across all or a subset of the layers constitute a shared community, and (b) the nodes form a community only in a specific layer and this community is not present in other layers. Applying these observations to the DeepWalk algorithm, we incorporate the following criteria while generating the random walks: For a shared community, the walks should encompass nodes located in different layers where the community resides. Whereas for a layer-specific community, the walks should be constrained in the same layer so that the learned representations will not be confused with communities in other layers. In order to achieve this, we introduce inter-layer edges based on the similarities between local community structures nodes. For a set of nodes $v_i$ and $v_j$ belonging to the layers $m$ and $n$ respectively, we construct inter-layer edges as follows:

$$\begin{aligned} E_{ij}^{(m)(n)} &= 0, i \neq j, \\ E_{ii}^{(m)(n)} &= \frac{|N_i^{(m)} \cap N_i^{(n)}|}{|N_i^{(m)} \cup N_i^{(n)}|}, \end{aligned} \tag{7.2}$$

where $E_{ij}^{(m)(n)}$ denotes the edge weight between node $i$ of layer $m$ and node $j$ of layer $n$. $N_i^{(m)}, N_i^{(n)}$ are the neighborhoods of node $i$ at layer $m$ and node $i$ at layer $n$ respectively. Note that, the edge weight is computed as the Jaccard coefficient of the agreement in the neighbor list between the two layers. The intuition is that each node in a layer can only be connected to its counterpart in another layer and similar neighborhoods across layers is a strong indication that they may form a shared community. The random walks which are able to encompass nodes at different layers according to Equation (7.2) will exploit both shared and layer-specific communities automatically.

We illustrate the multilayer graph with locally adaptive inter-layer connections and its corresponding supra adjacency matrix in Figure 27. In the supra adjacency matrix,

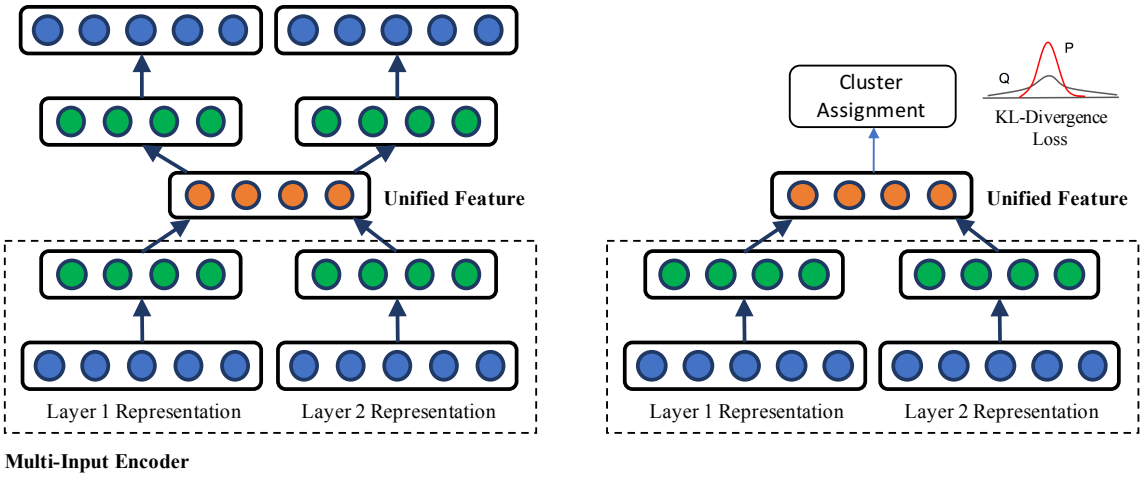



Figure 29: *Late fusion* - We construct a multi-input, multi-output autoencoder to obtain a unified feature space and fine-tune the encoder parameters based on a discriminative clustering objective. The resulting unified feature can then used with conventional clustering techniques to perform unsupervised tasks such as community detection.

each diagonal block matrix is the adjacency matrix corresponding to each of the layers. For each off-diagonal block, the diagonal entries have 1 where $E_{ii}^{(m)(n)}$ is larger than a certain threshold whereas all off-diagonal entries are all 0. Using the supra adjacency matrix defined in this way, we generate random walks for the DeepWalk algorithm using a uniform sampling strategy. Note that, the total number of nodes in the supra graph $M \times |\mathcal{V}|$ and hence produces $M$ different representations for each of the nodes. A simple concatenation of the latent representations corresponding to different layers serves as the popularly adopted form of late fusion. However, we propose a more powerful fusion approach that takes the underlying task of clustering into account. We will describe this fusion strategy in the next section.

In Figure 28, we demonstrate the behavior of the proposed early fusion strategy using the words layer in the Cora citation dataset. We visualize the latent spaces inferred for the citation and word layers using the DeepWalk algorithm before and

after introducing inter-layer edges to exploit the shared community structure. As it can be observed in the 2D t-SNE embeddings in Figure 28, the proposed early fusion strategy reveals cohesive communities in the graph by leveraging information across different layers. Consequently, the blue cluster in the *words* layer and the *cyan* cluster in the *citations* layer are more cohesive compared to their counterparts without the early fusion.

### 7.2.2 Late Fusion - Unified Deep Embedding for Clustering

Following the generation of latent features for the different layers using DeepWalk, we propose to employ a deep embedding technique for obtaining a unified representation for clustering tasks. This late fusion strategy is based on a recently proposed approach for deep clustering (Xie, Girshick, and Farhadi 2016), where the underlying idea is to adopt a data driven approach to infer the features and cluster assignments jointly. Such a joint optimization approach is particularly suited for our problem since we not only need to infer a unified representation for each of the nodes based on their latent features in each layer, but also attempt to ensure that the unified representation is optimal for the clustering task. In this approach, we first build a multi-input, multi-output autoencoder architecture to obtain the network parameters for constructing a unified feature space. Following this, we extract the multi-input encoder part of the network, fine-tune the parameters with an optimization objective similar to (Xie, Girshick, and Farhadi 2016). Figure 29 shows an overview of this approach.

First, in order to create a unified feature space for nodes in a multilayer graph, we build a multi-input, multi-output autoencoder where each input corresponds to the latent representation of each layer. As shown in Figure 29, a single input autoencoder is constructed for each feature set, outputs of which are merged using a concatenation

layer. In order to obtain a highly effective initialization, we first train a conventional single input autoencoder for each of the layers. Then the multi-input multi-output autoencoder is trained with the network parameters initialized from the previous training results.

After training the autoencoder, we discard the decoder part and fine-tune the parameters of the multi-input encoder for improved clustering performance. We adopt an approach similar to (Xie, Girshick, and Farhadi 2016) and it is comprised of two steps: (a) calculating the soft assignment probabilities and (b) jointly optimizing for both cluster centroids and feature embedding. We provide the algorithm outline here and the detailed description can be found in (Xie, Girshick, and Farhadi 2016).

Denoting the fusion and mapping function from $M$ feature sets to the unified embedding space by $f_\theta : \Phi^{(1)} \times \cdots \times \Phi^{(M)} \rightarrow \mathcal{Z}$, where $\Phi^{(1)}, \ldots, \Phi^{(M)}$ are the latent representations from $M$ layers of $\{G^{(m)}\}$ obtained using DeepWalk after early fusion, $\theta$ are the network parameters of the multi-input encoder and $\mathcal{Z}$ is the embedding space. We begin by estimating the clustering the nodes using the initial unified features and calculating similarity between the nodes and the cluster centroids in the unified embedding space using a strategy similar to stochastic neighborhood embedding algorithm (Xie, Girshick, and Farhadi 2016):

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + \|z_i - \mu_{j'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}} \tag{7.3}$$

These similarities can be interpreted as soft clustering assignments for the nodes. Assuming that the parameters $\theta$ are properly initialized and that this initialization is accurate in regions of low uncertainty, a reasonable clustering objective is to iteratively refine the clustering assignment by learning from the regions of low uncertainty. Interestingly, this is conceptually similar to discriminative clustering algorithms () that iteratively perform clustering (e.g. k-Means) to obtain cluster assignments

for data samples and supervised dimensionality reduction (e.g. linear discriminant analysis) based on current cluster labels to identify the discriminant function for maximal linear separability. In order to perform iterative, discriminative clustering with deep architectures, we adopt the optimization objective in (Xie, Girshick, and Farhadi 2016) that defines an auxiliary distribution, $p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}$, which is a peaked variant of the original distribution $q$. Here, $f_j = \sum_i q_{ij}$ are the soft cluster frequencies. Note, by constructing this auxiliary distribution, we identify a target distribution for the deep architecture to match, such that each node is assigned to one of the clusters with high certainty. Consequently, the joint optimization with respect to $\theta$ and $\mathcal{Z}$ is performed by minimizing the KL-divergence loss between the true distribution $p$ and the target $q$ for each of the nodes:

$$L = KL(P||Q) = \sum_i \sum_j p_{ij} \log(\frac{p_{ij}}{q_{ij}}) \tag{7.4}$$

The optimization is carried out using Stochastic Gradient Descent (SGD). After this step, the algorithm updates $q$ again according to Equation (7.3) and repeats this discriminative clustering procedure until convergence, in terms of cluster assignment.

## 7.3 Experimental Results

We begin by describing the datasets that were used to evaluate the proposed approach for inferencing from multilayer graphs. Subsequently, we will report results obtained using our method in comparison to the individual layers and conventional fusion strategies.

We consider a bibliographic datasets, Cora [5]. The Cora dataset consists of 2708 research papers focusing on machine learning from 7 different categories: case based, genetic algorithms, neural networks, probabilistic methods, reinforcement learning,

[5] www.cs.umd.edu/~sen/lbc-proj/LBC.html

Table 10: Clustering performance evaluation on the Cora citation dataset obtained using the proposed multilayer graph fusion strategies. The best results are marked in boldface.

| Graph Layers | DeepWalk | Early | Late | Accuracy | NMI | Purity |
|---|---|---|---|---|---|---|
| Words | N | N | N | 0.33 | 0.13 | 0.35 |
| Citations | N | N | N | 0.32 | 0.1 | 0.33 |
| Words | Y | N | N | 0.49 | 0.26 | 0.5 |
| Citations | Y | N | N | 0.42 | 0.32 | 0.52 |
| Words + Citations | Y | Y | N | 0.58 | 0.43 | 0.67 |
| Words + Citations | Y | Y | Y | **0.69** | **0.48** | **0.69** |

rule learning and theory. We used the category label as the ground truth for evaluating the clustering performance. In both pre-processed datasets, every paper cites or is cited by at least once. After the removal of trivial and infrequent words, Cora dataset has 1433 unique words.

We constructed two layers of graphs from data for our experiments: (a) Based on the binary features for word occurrence which indicates the presence (1) or absence (0) of each word, we calculated the pair-wise cosine similarity. Following common practice, the similarities are then converted to unweighted edges, i.e. binary edges, based on predefined thresholds; (b) We designed the citation layer describing the citation relation. If paper $a$ cites paper $b$ or paper $a$ is cited by paper $b$, then $a$ and $b$ are connected by an undirected edge in the graph.

We employed the proposed multilayer inferencing algorithm on the Cora citation dataset for clustering the nodes into meaningful communities. We then used the ground truth cluster labels to evaluate the clustering accuracy based on the three following metrics: accuracy, purity and normalized mutual information. As it can be observed in Table 10, the individual layers produce highly sub-optimal performance with features obtained using the conventional spectral embedding approach. When

we employed the DeepWalk algorithm to infer latent features for the two layers, the accuracies improve significantly. Incorporating the early fusion strategy leads to further improvements to both the layers. Note that in this case, we employed the standard late fusion strategy of simply concatenating the features prior to performing k-Means clustering. Finally including the late fusion strategy produces the best performance, as expected. The improvements in the inferencing performance can be directly attributed to the advanced fusion strategies that we employed.

# Chapter 8

# SUMMARY AND FUTURE WORK

## 8.1 Summary

In this dissertation, we discussed the representation learning based strategies in multimodal feature fusion. According to the application domain, we developed specific fusion solutions and demonstrated the improved performance as compared to state-of-the-art algorithms.

We described a novel approach for activity recognition by fusing multiple distinct features from multiple sensors. In particular, we presented a linearized variant of the multilayer graph consensus technique and effectively combined the discriminative capabilities of multiple sensors. Also we adopted a simple, reference-based classifier and fused the decisions from two distinct feature sets. We observed from our results that the framework can produce high quality recognition performances. Though we demonstrated our setup with this particular choice of sensors and features, the proposed two stage architecture is generally enough to be adapted to other applications as well.

By utilizing the kernel learning formulation, we presented a new approach for incorporating context modeling into low-level image kernel and showed that the fusion can be viewed as a MKL formulation. By building a series of classifiers to approximate the target posterior probabilities in the RKHS setting, we demonstrated improvements in the recognition performance. The presented framework is general enough to be adapted to other applications by utilizing appropriate context features.

Other kernel combination classifiers such as Multiple Kernel Logistic Regression (Kobayashi, Watanabe, and Otsu 2013) can also be incorporated to replace MKL.

We developed a principled approach to multimodal feature fusion by performing kernel learning using deep architectures. The proposed algorithm 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. By enabling the neural network to exploit the native space of a pre-defined 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. Using these improved representations, one can also perform multiple kernel learning efficiently. In addition to showing good convergence characteristics, the M-DKMO approach consistently outperforms 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.

We described two novel temporal modeling approaches for multivariate sequence analysis. The proposed approach to model clinical time-series data is solely based on masked self-attention and dispenses recurrence completely. Further, temporal order is incorporated into the sequence representation using both positional encoding and dense interpolation embedding techniques. The training process is efficient and the representations are highly effective for a wide-range of clinical diagnosis tasks. This is evidenced by the superior performance on the challenging MIMIC-III benchmark datasets. The second developed architecture studies the role of learning embeddings under a triplet ranking loss for speaker diarization. By utilizing a similar attention architecture, we combined the two steps of GMM-UBM training and a separate triplet

training into a single end-to-end model. Results on the CALLHOME corpus show that this joint training process achieves improved diarization performance with less training efforts.

In order to tackle graph-structured data for multimodal representation learning, we extended the DeepWalk node embedding algorithm to multilayer graphs and developed systematic fusion procedures which incorporate both intralayer and interlayer information. Focusing on the community detection problem, we demonstrated improvement partitioning capability of the proposed algorithm as compared to straightforward fusion methods.

## 8.2 Future Work

There are abundant opportunities to further extend the described fusion algorithms. We describe several promising directions as future work:

In the DKMO and M-DKMO approach, we focused on utilizing deep learning to optimize kernel machines. From another viewpoint, similar to the recent approaches such as the convolutional kernel networks (Mairal 2016), principles from kernel learning theory can enable the design of novel training strategies for neural networks. In particular, incorporating kernel formulations into the deep architecture itself could provide better regularization of the DNN learning. Potentially, this can be 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.

Attention networks are recently proposed approaches for sequence to sequence problem in NLP. Based the findings in our works for clinical time series and speech data, attention networks have huge potential in a wide range of temporal modeling

applications. Moving forward, we plan to investigate the role of attentions in Generative Adversarial Networks (GANs) and in multilayer graph mining.

Due to the sensitive nature of clinical time series data, its access is typically very restricted and carefully controlled. This prohibits the successful adoption of deep learning models, which require large amount of data to learn complex structures without overfitting. In order to overcome this limitation, very recent research (Esteban, Hyland, and Rätsch 2017) proposes to use Generative Adversarial Networks (GANs) to generate realistic synthetic medical data. GANs have shown remarkable success in producing realistic-looking images and in order to adopt them to multivariate time-series data, (Esteban, Hyland, and Rätsch 2017) substituted both the generator and discriminator with LSTMs. Inspired by the success of attention models in clinical modeling, we are interested in how attentions can be used in the synthesis of clinical time series with GANs. Besides replacing LSTM RNNs with attention models, new training strategies and special handling of the matrix representations need to be investigated.

Recently, attention models have been shown to be effective on graph-structured data as well. As reported in (Velicković et al. 2017), leveraging the self-attention mechanism enables implicitly specifying different weights for nodes in a neighborhood. Similar to the problem we have addressed in Chapter 7, multiple relations often exist in real-world applications and in this scenario, interlayer attention needs to be properly derived to construct a predictive model on multilayer graphs.

In terms of speech modeling including speaker diarization, we are interested in investigating new metric learning techniques and applications on speech. Specifically, future work will investigate more sophisticated sampling strategies and novel loss

functions under the siamese and triplet network settings (Manmatha et al. 2017; Chen et al. 2017).

# REFERENCES


Abu-El-Haija, Sami, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. 2016. "Youtube-8m: A large-scale video classification benchmark." *arXiv preprint arXiv:1609.08675.*

Agarwal, Gaurav, and David Kempe. 2008. "Modularity-maximizing graph communities via mathematical programming." *The European Physical Journal B* 66 (3): 409–418.

Akaike, H. 1974. "A new look at the statistical model identification." *Automatic Control, IEEE Transactions on* 19 (6): 716–723.

Almeida, Hélio, Dorgival Guedes, Wagner Meira, and Mohammed J Zaki. 2011. "Is there a best quality metric for graph clusters?" In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases,* 44–59. Springer.

Andreeva, Antonina, Dave Howorth, Cyrus Chothia, Eugene Kulesha, and Alexey G Murzin. 2014. "SCOP2 prototype: a new approach to protein structure mining." *Nucleic acids research* 42 (D1): D310–D314.

Andrew, Alex M. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods by Nello Christianini and John Shawe-Taylor, Cambridge University Press, Cambridge, 2000, xiii+ 189 pp., ISBN 0-521-78019-5 (Hbk,£ 27.50).*

Andrew, Galen, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. "Deep canonical correlation analysis." In *International Conference on Machine Learning,* 1247–1255.

Avci, A., S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. 2010. "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey." In *ARCS, 2010 23rd international conference on,* 1–10. VDE.

Bach, Francis R, Gert RG Lanckriet, and Michael I Jordan. 2004. "Multiple kernel learning, conic duality, and the SMO algorithm." In *Proceedings of the twenty-first international conference on Machine learning,* 6. ACM.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473.*

Baltrušaitis, Tadas, Chaitanya Ahuja, and Louis-Philippe Morency. 2017. "Multimodal Machine Learning: A Survey and Taxonomy." *arXiv preprint arXiv:1705.09406.*

Bashivan, Pouya, Irina Rish, Mohammed Yeasin, and Noel Codella. 2015. "Learning representations from EEG with deep recurrent-convolutional neural networks." *arXiv preprint arXiv:1511.06448.*

Berlingerio, Michele, Michele Coscia, and Fosca Giannotti. 2011. "Finding and characterizing communities in multidimensional networks." In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on,* 490–494. IEEE.

Berlingerio, Michele, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. 2011. "Foundations of multidimensional network analysis." In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on,* 485–489. IEEE.

Berlingerio, Michele, Fabio Pinelli, and Francesco Calabrese. 2013. "Abacus: frequent pattern mining-based community discovery in multidimensional networks." *Data Mining and Knowledge Discovery* 27 (3): 294–320.

Bhagat, Smriti, Graham Cormode, and S Muthukrishnan. 2011. "Node classification in social networks." In *Social network data analytics,* 115–148. Springer.

Bishop, Christopher M. 2006. "Machine learning and pattern recognition." *Information Science and Statistics. Springer, Heidelberg.*

Blondel, Vincent D, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. "Fast unfolding of communities in large networks." *Journal of statistical mechanics: theory and experiment* 2008 (10): P10008.

Boutemine, Oualid, and Mohamed Bouguessa. 2017. "Mining community structures in multidimensional networks." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11 (4): 51.

Bredin, Hervé. 2017a. "pyannote. metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems." In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association.*

———. 2017b. "Tristounet: triplet loss for speaker turn embedding." In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on,* 5430–5434. IEEE.

Bucak, Serhat S, Rong Jin, and Anil K Jain. 2014. "Multiple kernel learning for visual object recognition: A review." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (7): 1354–1369.

Cavallari, Sandro, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. 2017. "Learning community embedding with community detection and node embedding on graphs." In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management,* 377–386. ACM.

Chang, Chih-Chung, and Chih-Jen Lin. 2011. "LIBSVM: a library for support vector machines." *ACM Transactions on Intelligent Systems and Technology (TIST)* 2 (3): 27.

Chapelle, Olivier, and Alexander Zien. 2005. "Semi-Supervised Classification by Low Density Separation." In *AISTATS,* 57–64.

Che, Zhengping, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2016. "Recurrent neural networks for multivariate time series with missing values." *arXiv preprint arXiv:1606.01865.*

Chen, H.-Tzong, H.-Wei Chang, and T.-Luh Liu. 2005. "Local discriminant embedding and its variants." In *CVPR 2005. IEEE Computer Society Conference on,* 2:846–853. IEEE.

Chen, Mingming, Konstantin Kuzmin, and Boleslaw K Szymanski. 2014. "Community detection via maximization of modularity and its variants." *IEEE Transactions on Computational Social Systems* 1 (1): 46–65.

Chen, Weihua, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. 2017. "Beyond triplet loss: a deep quadruplet network for person re-identification." In *Proc. CVPR,* vol. 2.

Cho, Y., and L. K. Saul. 2009. "Kernel methods for deep learning." In *Advances in neural information processing systems,* 342–350.

Chollet, François, et al. 2015. *Keras.* https://github.com/fchollet/keras.

Chopra, Sumit, Raia Hadsell, and Yann LeCun. 2005. "Learning a similarity metric discriminatively, with application to face verification." In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on,* 1:539–546. IEEE.

Cook, Diane J., and Lawrence B. Holder. 2006. *Mining Graph Data.* John Wiley & Sons.

Correa, Nicolle M, Tulay Adali, Yi-Ou Li, and Vince D Calhoun. 2010. "Canonical correlation analysis for data fusion and group inferences." *IEEE signal processing magazine* 27 (4): 39–50.

Cortes, Corinna, Mehryar Mohri, and Afshin Rostamizadeh. 2009. "Learning non-linear combinations of kernels." In *Advances in neural information processing systems,* 396–404.

———. 2013. "Multi-Class Classification with Maximum Margin Multiple Kernel." In *ICML (3),* 46–54.

Cui, Yiming, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. "Attention-over-attention neural networks for reading comprehension." *arXiv preprint arXiv:1607.04423.*

Dalal, Navneet, and Bill Triggs. 2005. "Histograms of oriented gradients for human detection." In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on,* 1:886–893. IEEE.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. "Imagenet: A large-scale hierarchical image database." In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on,* 248–255. IEEE.

Dhillon, Paramveer, Dean P Foster, and Lyle H Ungar. 2011. "Multi-view learning of word embeddings via cca." In *Advances in Neural Information Processing Systems,* 199–207.

Ding, Chris HQ, and Inna Dubchak. 2001. "Multi-class protein fold recognition using support vector machines and neural networks." *Bioinformatics* 17 (4): 349–358.

Dong, Xiaowen, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. 2012. "Clustering with multi-layer graphs: A spectral perspective." *IEEE Transactions on Signal Processing* 60 (11): 5820–5831.

Drineas, Petros, and Michael W Mahoney. 2005. "On the Nyström method for approximating a Gram matrix for improved kernel-based learning." *journal of machine learning research* 6 (Dec): 2153–2175.

Duch, Jordi, and Alex Arenas. 2005. "Community detection in complex networks using extremal optimization." *Physical review E* 72 (2): 027104.

Eagle, Nathan, and Alex Sandy Pentland. 2006. "Reality mining: sensing complex social systems." *Personal and ubiquitous computing* 10 (4): 255–268.

Esteban, Cristóbal, Stephanie L Hyland, and Gunnar Rätsch. 2017. "Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs." *arXiv preprint arXiv:1706.02633.*

Fernando, Basura, Elisa Fromont, Damien Muselet, and Marc Sebban. 2012. "Discriminative feature fusion for image classification." In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on,* 3434–3441. IEEE.

Fox, Emily B, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. 2011. "A sticky HDP-HMM with application to speaker diarization." *The Annals of Applied Statistics:* 1020–1056.

Frank, J., S. Mannor, and D. Precup. 2010. "Activity and Gait Recognition with Time-Delay Embeddings." In *AAAI.* Citeseer.

Fukui, Akira, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. "Multimodal compact bilinear pooling for visual question answering and visual grounding." *arXiv preprint arXiv:1606.01847.*

Gales, Mark, Steve Young, et al. 2008. "The application of hidden Markov models in speech recognition." *Foundations and Trends® in Signal Processing* 1 (3): 195–304.

Gao, Yang, Oscar Beijbom, Ning Zhang, and Trevor Darrell. 2016. "Compact bilinear pooling." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 317–326.

Garcia-Romero, Daniel, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree. 2017. "Speaker diarization using deep neural network embeddings." In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on,* 4930–4934. IEEE.

Gehler, Peter, and Sebastian Nowozin. 2009. "On feature combination for multiclass object classification." In *Computer Vision, 2009 IEEE International Conference on,* 221–228. IEEE.

Ghassemi, Marzyeh, Marco AF Pimentel, Tristan Naumann, Thomas Brennan, David A Clifton, Peter Szolovits, and Mengling Feng. 2015. "A Multivariate Timeseries Modeling Approach to Severity of Illness Assessment and Forecasting in ICU with Sparse, Heterogeneous Clinical Data." In *AAAI,* 446–453.

Gligorijević, Vladimir, Yannis Panagakis, and Stefanos Zafeiriou. 2016. "Fusion and community detection in multi-layer graphs." In *Pattern Recognition (ICPR), 2016 23rd International Conference on,* 1327–1332. IEEE.

Gönen, Mehmet, and Ethem Alpaydin. 2008. "Localized multiple kernel learning." In *Proceedings of the 25th international conference on Machine learning,* 352–359. ACM.

Gönen, Mehmet, and Ethem Alpaydın. 2011. "Multiple kernel learning algorithms." *The Journal of Machine Learning Research* 12:2211–2268.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. "Speech recognition with deep recurrent neural networks." In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on,* 6645–6649. IEEE.

Gravina, Raffaele, Parastoo Alinia, Hassan Ghasemzadeh, and Giancarlo Fortino. 2017. "Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges." *Information Fusion* 35:68–80.

Grover, Aditya, and Jure Leskovec. 2016. "node2vec: Scalable feature learning for networks." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 855–864. ACM.

Guo, Zhiqiang, Huaiqing Wang, Quan Liu, and Jie Yang. 2014. "A feature fusion based forecasting model for financial time series." *PloS one* 9 (6): e101113.

Hamilton, Will, Zhitao Ying, and Jure Leskovec. 2017. "Inductive representation learning on large graphs." In *Advances in Neural Information Processing Systems,* 1025–1035.

Harris, Z.S. 1954. *Distributional Structure.* https://books.google.com/books?id=ycMmcgAACAAJ.

Harutyunyan, Hrayr, Hrant Khachatrian, David C Kale, and Aram Galstyan. 2017. "Multitask Learning and Benchmarking with Clinical Time Series Data." *arXiv preprint arXiv:1703.07771.*

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep residual learning for image recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 770–778.

He, Z.-Yu, and L.-Wen Jin. 2008. "Activity recognition from acceleration data using AR model representation and SVM." In *Machine Learning and Cybernetics, 2008 International Conference on,* 4:2245–2250. IEEE.

Hermann, Karl Moritz, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. "Teaching machines to read and comprehend." In *Advances in Neural Information Processing Systems,* 1693–1701.

Hinton, Geoffrey E, and Ruslan R Salakhutdinov. 2006. "Reducing the dimensionality of data with neural networks." *science* 313 (5786): 504–507.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long short-term memory." *Neural computation* 9 (8): 1735–1780.

Hoffer, Elad, and Nir Ailon. 2015. "Deep metric learning using triplet network." In *International Workshop on Similarity-Based Pattern Recognition,* 84–92. Springer.

Hong, Dug Hun, and Chang-Hwan Choi. 2000. "Multicriteria fuzzy decision-making problems based on vague set theory." *Fuzzy sets and systems* 114 (1): 103–113.

Huang, Po-Sen, Haim Avron, Tara N. Sainath, Vikas Sindhwani, and Bhuvana Ramabhadran. 2014. "Kernel methods match Deep Neural Networks on TIMIT." In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on,* 205–209.

Ioffe, Sergey, and Christian Szegedy. 2015. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In *Proceedings of The 32nd International Conference on Machine Learning,* 448–456.

Jain, A., S. VN Vishwanathan, and M. Varma. 2012. "SPG-GMKL: generalized multiple kernel learning with a million kernels." In *18th KDD,* 750–758. ACM.

Jhuo, I-Hong, and DT Lee. 2010. "Boosted multiple kernel learning for scene category recognition." In *Pattern Recognition (ICPR), 2010 20th International Conference on,* 3504–3507. IEEE.

Jia, Y.Q., F.P. Nie, and C.S. Zhang. 2009. "Trace ratio problem revisited." *Neural Networks, IEEE Transactions on* 20 (4): 729–735.

Joachims, Thorsten. 2003. "Transductive learning via spectral graph partitioning." In *Proceedings of the 20th International Conference on Machine Learning (ICML-03),* 290–297.

Johnson, Alistair EW, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. "MIMIC-III, a freely accessible critical care database." *Scientific data* 3.

Kannao, R., and P. Guha. 2016. "TV Commercial Detection Using Success Based Locally Weighted Kernel Combination." In *MultiMedia Modeling,* 793–805. Springer.

Karpathy, Andrej, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. "Large-scale video classification with convolutional neural networks." In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition,* 1725–1732.

Kaufman, Leonard, and Peter J Rousseeuw. 2009. *Finding groups in data: an introduction to cluster analysis.* Vol. 344. John Wiley & Sons.

Kelly, AM Clare, Lucina Q Uddin, Bharat B Biswal, F Xavier Castellanos, and Michael P Milham. 2008. "Competition between functional brain networks mediates behavioral variability." *Neuroimage* 39 (1): 527–537.

Khoury, Elie, Laurent El Shafey, Marc Ferras, and Sébastien Marcel. 2014. "Hierarchical speaker clustering methods for the NIST i-vector Challenge." In *Odyssey: The Speaker and Language Recognition Workshop.*

Kim, Jungeun, and Jae-Gil Lee. 2015. "Community detection in multi-layer graphs: A survey." *ACM SIGMOD Record* 44 (3): 37–48.

Kim, Jungeun, Jae-Gil Lee, and Sungsu Lim. 2017. "Differential flattening: A novel framework for community detection in multi-layer graphs." *ACM Transactions on Intelligent Systems and Technology (TIST)* 8 (2): 27.

Kim, Yoon. 2014. "Convolutional neural networks for sentence classification." *arXiv preprint arXiv:1408.5882.*

Kingma, Diederik, and Jimmy Ba. 2014. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980.*

Kipf, Thomas N, and Max Welling. 2016. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907.*

Kivelä, Mikko, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. 2014. "Multilayer networks." *Journal of complex networks* 2 (3): 203–271.

Kobayashi, Takumi, Kenji Watanabe, and Nobuyuki Otsu. 2013. "Revisit of Logistic Regression: Efficient Optimization and Kernel Extensions." *International Journal of Advanced Computer Science and Applications* 4 (5): 138–147.

Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. 2015. "Siamese neural networks for one-shot image recognition." In *ICML Deep Learning Workshop,* vol. 2.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems,* 1097–1105.

Kumar, Sanjiv, Mehryar Mohri, and Ameet Talwalkar. 2009. "Ensemble nystrom method." In *Advances in Neural Information Processing Systems,* 1060–1068.

———. 2012. "Sampling methods for the Nyström method." *Journal of Machine Learning Research* 13 (Apr): 981–1006.

Kun, Jeremy, Rajmonda Caceres, and Kevin Carter. 2014. "Locally boosted graph aggregation for community detection." *arXiv preprint arXiv:1405.3210.*

Kuncheva, Zhana, and Giovanni Montana. 2015. "Community detection in multiplex networks using locally adaptive random walks." In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015,* 1308–1315. ACM.

Kwapisz, J. R, G. M Weiss, and S. A Moore. 2011. "Activity recognition using cell phone accelerometers." *ACM SigKDD Explorations Newsletter* 12 (2): 74–82.

Lai, Pei Ling, and Colin Fyfe. 2000. "Kernel and nonlinear canonical correlation analysis." *International Journal of Neural Systems* 10 (05): 365–377.

Le Lan, Gaël, Delphine Charlet, Anthony Larcher, and Sylvain Meignier. 2017. "A Triplet Ranking-based Neural Network for Speaker Diarization and Linking." *Proc. Interspeech 2017:* 3572–3576.

Le, Quoc, Tamas Sarlos, and Alex Smola. 2013. "Fastfood - Approximating Kernel Expansions in Loglinear Time." In *30th International Conference on Machine Learning (ICML).* http://jmlr.org/proceedings/papers/v28/le13.html.

Levy, O., and Y. Goldberg. 2014. "Neural word embedding as implicit matrix factorization." In *Advances in neural information processing systems,* 2177–2185.

Li, Mu, Wei Bi, James T Kwok, and Bao-Liang Lu. 2015. "Large-scale Nyström kernel matrix approximation using randomized SVD." *IEEE transactions on neural networks and learning systems* 26 (1): 152–164.

Li, Q., H.G. Zhang, J. Guo, B. Bhanu, and L. An. 2013. "Reference-based scheme combined with k-svd for scene image categorization." *Signal Processing Letters, IEEE* 20 (1): 67–70.

Liben-Nowell, David, and Jon Kleinberg. 2007. "The link-prediction problem for social networks." *journal of the Association for Information Science and Technology* 58 (7): 1019–1031.

Lin, Tsung-Yi, Yin Cui, Serge Belongie, and James Hays. 2015. "Learning deep representations for ground-to-aerial geolocalization." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 5007–5015.

Lin, Tsung-Yu, Aruni RoyChowdhury, and Subhransu Maji. 2015. "Bilinear cnn models for fine-grained visual recognition." In *Proceedings of the IEEE International Conference on Computer Vision,* 1449–1457.

Lipton, Zachary C, David C Kale, Charles Elkan, and Randall Wetzell. 2015. "Learning to diagnose with LSTM recurrent neural networks." *arXiv preprint arXiv:1511.03677.*

Lipton, Zachary C, David C Kale, and Randall Wetzel. 2016. "Modeling Missing Data in Clinical Time Series with RNNs." *arXiv preprint arXiv:1606.04130.*

Liu, Fayao, Luping Zhou, Chunhua Shen, and Jianping Yin. 2014. "Multiple kernel learning in the primal for multimodal Alzheimer's disease classification." *IEEE journal of biomedical and health informatics* 18 (3): 984–990.

Liu, Kui, Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. 2014. "Fusion of inertial and depth sensor data for robust hand gesture recognition." *IEEE Sensors Journal* 14 (6): 1898–1903.

Liu, Weiyi, Pin-yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. 2017. "Principled Multilayer Network Embedding." In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on,* 134–141. IEEE.

Liu, Zitao, and Milos Hauskrecht. 2013. "Clinical time series prediction with a hierarchical dynamical system." In *Conference on Artificial Intelligence in Medicine in Europe,* 227–237. Springer.

Lodhi, Huma, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. "Text classification using string kernels." *Journal of Machine Learning Research* 2 (Feb): 419–444.

Lowe, David G. 1999. "Object recognition from local scale-invariant features." In *Computer Vision, 1999 IEEE International Conference on,* 2:1150–1157. IEEE.

Maaten, Laurens van der, and Geoffrey Hinton. 2008. "Visualizing data using t-SNE." *Journal of Machine Learning Research* 9 (Nov): 2579–2605.

Magnani, Matteo, Barbora Micenkova, and Luca Rossi. 2013. "Combinatorial analysis of multiple networks." *arXiv preprint arXiv:1303.4986.*

Mairal, Julien. 2016. "End-to-end kernel learning with supervised convolutional kernel networks." In *Advances in Neural Information Processing Systems,* 1399–1407.

Mairal, Julien, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. 2014. "Convolutional kernel networks." In *Advances in Neural Information Processing Systems,* 2627–2635.

Manmatha, R, Chao-Yuan Wu, Alexander J Smola, and Philipp Krähenbühl. 2017. "Sampling Matters in Deep Embedding Learning." In *Computer Vision (ICCV), 2017 IEEE International Conference on,* 2859–2867. IEEE.

Martinez, Hector P, and Georgios N Yannakakis. 2014. "Deep multimodal fusion: Combining discrete events and continuous signals." In *Proceedings of the 16th International conference on multimodal interaction,* 34–41. ACM.

Mikolov, Tomas, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. "Recurrent neural network based language model." In *Interspeech,* 2:3.

Moeller, John, Sarathkrishna Swaminathan, and Suresh Venkatasubramanian. 2016. "A Unified View of Localized Kernel Learning." In *Proceedings of the 2016 SIAM International Conference on Data Mining,* 252–260. SIAM.

Monti, Federico, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. "Geometric deep learning on graphs and manifolds using mixture model CNNs." In *Proc. CVPR,* 1:3. 2.

Mucha, Peter J, Thomas Richardson, Kevin Macon, Mason A Porter, and Jukka-Pekka Onnela. 2010. "Community structure in time-dependent, multiscale, and multiplex networks." *science* 328 (5980): 876–878.

Nair, Vinod, and Geoffrey E Hinton. 2010. "Rectified linear units improve restricted boltzmann machines." In *Proceedings of the 27th international conference on machine learning (ICML-10),* 807–814.

Neverova, Natalia, Christian Wolf, Graham Taylor, and Florian Nebout. 2016. "Moddrop: adaptive multi-modal gesture recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38 (8): 1692–1706.

Newman, Mark EJ. 2006. "Finding community structure in networks using the eigenvectors of matrices." *Physical review E* 74 (3): 036104.

Ng, Andrew Y, Michael I Jordan, Yair Weiss, et al. 2001. "On spectral clustering: Analysis and an algorithm." In *NIPS,* 14:849–856. 2.

Ngiam, J., A. Khosla, M. Kim, J. Nam, H. Lee, and A. Ng. 2011. "Multimodal deep learning." In *Proceedings of the 28th ICML,* 689–696.

Nilsback, Maria-Elena, and Andrew Zisserman. 2008. "Automated flower classification over a large number of classes." In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on,* 722–729. IEEE.

Ong, Cheng Soon, and Alexander Zien. 2008. "An automated combination of kernels for predicting protein subcellular localization." In *International Workshop on Algorithms in Bioinformatics,* 186–197. Springer.

Orabona, Francesco, and Luo Jie. 2011. "Ultra-fast optimization algorithm for sparse multi kernel learning." In *Proceedings of the 28th International Conference on Machine Learning (ICML-11),* 249–256.

Orabona, Francesco, Luo Jie, and Barbara Caputo. 2012. "Multi kernel learning with online-batch optimization." *Journal of Machine Learning Research* 13 (Feb): 227–253.

Orwat, C., A. Graefe, and T. Faulwasser. 2008. "Towards pervasive computing in health care–A literature review." *BMC Medical Informatics and Decision Making* 8 (1): 26.

Papalexakis, Evangelos E, Leman Akoglu, and Dino Ience. 2013. "Do more views of a graph help? community detection and clustering in multi-graphs." In *Information fusion (FUSION), 2013 16th international conference on,* 899–905. IEEE.

Park, Eunbyung, Xufeng Han, Tamara L Berg, and Alexander C Berg. 2016. "Combining multiple sources of knowledge in deep cnns for action recognition." In

*Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on,* 1–8. IEEE.

Paulus, Romain, Caiming Xiong, and Richard Socher. 2017. "A Deep Reinforced Model for Abstractive Summarization." *arXiv preprint arXiv:1705.04304.*

Pelleg, Dan, Andrew W Moore, et al. 2000. "X-means: Extending k-means with efficient estimation of the number of clusters." In *Icml,* 1:727–734.

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. 2014. "Deepwalk: Online learning of social representations." In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining,* 701–710. ACM.

Polikar, Robi. 2006. "Ensemble based systems in decision making." *IEEE Circuits and systems magazine* 6 (3): 21–45.

Qi, Xianbiao, Rong Xiao, Chun-Guang Li, Yu Qiao, Jun Guo, and Xiaoou Tang. 2014. "Pairwise rotation invariant co-occurrence local binary pattern." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (11): 2199–2213.

Rabiner, Lawrence R. 1989. "A tutorial on hidden Markov models and selected applications in speech recognition." *Proceedings of the IEEE* 77 (2): 257–286.

Rahimi, Ali, and Benjamin Recht. 2008. "Random Features for Large-Scale Kernel Machines." In *Advances in Neural Information Processing Systems 20,* 1177–1184.

Rakotomamonjy, Alain, Francis R Bach, Stéphane Canu, and Yves Grandvalet. 2008. "SimpleMKL." *Journal of Machine Learning Research* 9 (Nov): 2491–2521.

Ramamurthy, Karthikeyan, Jayaraman Jayaraman Thiagarajan, Prasanna Sattigeri, and Andreas Spanias. 2016. *Ensemble sparse models for image analysis and restoration.* US Patent App. 14/772,343, January.

Ramamurthy, K.N., J.J. Thiagarajan, R. Sridhar, P. Kothandaraman, and R. Nachiappan. 2014. "Consensus inference with multilayer graphs for multi-modal data." In *Signals, Systems and Computers, 2014 48th Asilomar Conference on,* 1341–1345. November. doi:10.1109/ACSSC.2014.7094679.

Reynolds, Douglas A. 1995. "Speaker identification and verification using Gaussian mixture speaker models." *Speech communication* 17 (1-2): 91–108.

Ryali, Srikanth, Kaustubh Supekar, Tianwen Chen, and Vinod Menon. 2011. "Multivariate dynamical systems models for estimating causal interactions in fMRI." *Neuroimage* 54 (2): 807–823.

Sak, Haşim, Andrew Senior, and Françoise Beaufays. 2014. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." In *Fifteenth annual conference of the international speech communication association.*

Sauer, T., J. A Yorke, and M. Casdagli. 1991. "Embedology." *Journal of statistical Physics* 65 (3-4): 579–616.

Scholkopf, Bernhard, and Alexander J. Smola. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press.

Schölkopf, Bernhard, and Alexander J Smola. 2002. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin. 2015. "Facenet: A unified embedding for face recognition and clustering." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 815–823.

Sell, Gregory, and Daniel Garcia-Romero. 2014. "Speaker diarization with PLDA i-vector scoring and unsupervised calibration." In *Spoken Language Technology Workshop (SLT), 2014 IEEE,* 413–417. IEEE.

Shuman, D. I., S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. 2013. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains." *IEEE Signal Processing Magazine* 30, no. 3 (May): 83–98.

Simonyan, Karen, and Andrew Zisserman. 2014. "Two-stream convolutional networks for action recognition in videos." In *Advances in neural information processing systems,* 568–576.

Song, H., J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias. 2016. "Auto-context modeling using multiple Kernel learning." In *IEEE ICIP,* 1868–1872. Phoenix: IEEE, September.

Song, H., J. J. Thiagarajan, K. N. Ramamurthy, A. Spanias, and P. Turaga. 2016. "Consensus inference on mobile phone sensors for activity recognition." In *IEEE ICASSP,* 2294–2298. Shanghai: IEEE, March.

Song, H., J. J. Thiagarajan, P. Sattigeri, K. N. Ramamurthy, and A. Spanias. 2017. "A deep learning approach to multiple kernel fusion." In *IEEE ICASSP.* New Orleans: IEEE, March.

Song, Huan, Deepta Rajan, Jayaraman J Thiagarajan, and Andreas Spanias. 2018. "Attend and Diagnose: Clinical Time Series Analysis using Attention Models." In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-2018).* New Orleans, March.

Song, Huan, Jayaraman J Thiagarajan, Prasanna Sattigeri, and Andreas Spanias. 2018. "Optimizing Kernel Machines Using Deep Learning." *IEEE Transactions on Neural Networks and Learning Systems.*

Song, Huan, Megan Willi, Jayaraman J. Thiagarajan, Visar Berisha, and Andreas Spanias. 2018. "Triplet Network with Attention for Speaker Diarization." *Proc. Interspeech 2018.*

Spanias, A. 2014. *Digital Signal Processing: An Interactive Approach.* ISBN 978-1-4675-9892-7, Morrisville, NC: Lulu Press On-demand Publishers.

Srinivas, Suraj, Ravi Kiran Sarvadevabhatla, Konda Reddy Mopuri, Nikita Prabhu, Srinivas SS Kruthiventi, and R Venkatesh Babu. 2016. "A taxonomy of deep convolutional neural nets for computer vision." *arXiv preprint arXiv:1601.06615.*

Srivastava, N., G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research* 15 (1): 1929–1958.

Srivastava, Nitish, and Ruslan R Salakhutdinov. 2012. "Multimodal learning with deep boltzmann machines." In *Advances in neural information processing systems,* 2222–2230.

Sumway, Robert H, and David S Stoffer. 2006. "Time series analysis and its applications with R examples." *Time series analysis and its applications with R examples.*

Sun, Zhaonan, Nawanol Ampornpunt, Manik Varma, and Svn Vishwanathan. 2010. "Multiple kernel learning and the SMO algorithm." In *Advances in neural information processing systems,* 2361–2369.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le. 2014. "Sequence to sequence learning with neural networks." In *Advances in neural information processing systems,* 3104–3112.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 1–9.

Tagarelli, Andrea, Alessia Amelio, and Francesco Gullo. 2017. "Ensemble-based community detection in multilayer networks." *Data Mining and Knowledge Discovery* 31 (5): 1506–1543.

Tang, Wei, Zhengdong Lu, and Inderjit S Dhillon. 2009. "Clustering with multiple graphs." In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on,* 1016–1021. IEEE.

Thiagarajan, Jayaraman J, Peer-Timo Bremer, and Karthikeyan Natesan Ramamurthy. 2014. "Multiple kernel interpolation for inverting non-linear dimensionality reduction and dimension estimation." In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on,* 6751–6755.

Thiagarajan, J.J., K.N. Ramamurthy, P. Turaga, and A. Spanias. 2014. *Image Understanding Using Sparse Representations.* 7:1–118. 1. Morgan & Claypool Publishers.

Trask, Andrew, David Gilmore, and Matthew Russell. 2015. "Modeling order in neural word embeddings at scale." *arXiv preprint arXiv:1506.02338.*

Tsuda, Koji, Taishin Kin, and Kiyoshi Asai. 2002. "Marginalized kernels for biological sequences." *Bioinformatics* 18 (suppl 1): S268–S275.

Tu, Zhuowen. 2008. "Auto-context and its application to high-level vision tasks." In *Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on,* 1–8. IEEE.

Van De Weijer, Joost, and Cordelia Schmid. 2006. "Coloring local feature extraction." In *Computer Vision–ECCV 2006,* 334–348. Springer.

Van Der Maaten, Laurens. 2014. "Accelerating t-SNE using tree-based algorithms." *Journal of machine learning research* 15 (1): 3221–3245.

Varga, A_P, and RK Moore. 1990. "Hidden Markov model decomposition of speech and noise." In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on,* 845–848. IEEE.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention is all you need." In *Advances in Neural Information Processing Systems,* 6000–6010.

Velicković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lió, and Yoshua Bengio. 2017. "Graph Attention Networks." *arXiv preprint arXiv:1710.10903.*

Venkataraman, V., and P. Turaga. 2016. "Shape Descriptions of Nonlinear Dynamical Systems for Video-based Inference." *IEEE Trans. on Pattern Analysis and Machine Intelligence* PP (99): 1–1.

Venkataraman, Vinay, Pavan Turaga, Nicole Lehrer, Michael Baran, Thanassis Rikakis, and Steven Wolf. 2013. "Attractor-shape for dynamical analysis of human movement: Applications in stroke rehabilitation and action recognition." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops,* 514–520.

Vinyals, Oriol, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. "Grammar as a foreign language." In *Advances in Neural Information Processing Systems,* 2773–2781.

Vishwanathan, S Vichy N, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. "Graph kernels." *Journal of Machine Learning Research* 11 (Apr): 1201–1242.

Vlachos, I, and D Kugiumtzis. 2008. "State space reconstruction for multivariate time series prediction." *arXiv preprint arXiv:0809.2220.*

Von Luxburg, Ulrike. 2007. "A tutorial on spectral clustering." *Statistics and computing* 17 (4): 395–416.

Wang, Quan, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopez Moreno. 2017. "Speaker diarization with lstm." *arXiv preprint arXiv:1710.10468.*

Wang, Xiao, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. "Community Preserving Network Embedding." In *AAAI,* 203–209.

Wiering, Marco A, and Lambert RB Schomaker. 2014. "Multi-layer support vector machines." In *Regularization, Optimization, Kernels, and Support Vector Machines,* 457–475. Chapman / Hall/CRC.

Wilson, Andrew Gordon, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. 2016. "Deep kernel learning." In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics,* 370–378.

Wu, Ting-fan, Chih-Jen Lin, and Ruby C. Weng. 2003. "Probability Estimates for Multi-class Classification by Pairwise Coupling." *Journal of Machine Learning Research* 5:975–1005.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. "Google's neural machine translation system: Bridging the gap between human and machine translation." *arXiv preprint arXiv:1609.08144.*

Xie, Junyuan, Ross Girshick, and Ali Farhadi. 2016. "Unsupervised deep embedding for clustering analysis." In *International Conference on Machine Learning (ICML).*

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. "Show, attend and tell: Neural image caption generation with visual attention." In *International Conference on Machine Learning,* 2048–2057.

Yang, Zhilin, William W Cohen, and Ruslan Salakhutdinov. 2016. "Revisiting semi-supervised learning with graph embeddings." *arXiv preprint arXiv:1603.08861.*

Ye, K., and L.H. Lim. 2014. "Distance between subspaces of different dimensions." *arXiv preprint arXiv:1407.0900.*

Yeh, Yi-Ren, Ting-Chu Lin, Yung-Yu Chung, and Yu-Chiang Frank Wang. 2012. "A novel multiple kernel learning framework for heterogeneous feature fusion and variable selection." *Multimedia, IEEE Transactions on* 14 (3): 563–574.

Zhang, Jianguo, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. 2007. "Local features and kernels for classification of texture and object categories: A comprehensive study." *International journal of computer vision* 73 (2): 213–238.

Zhang, Kai, and James T Kwok. 2010. "Clustered Nyström method for large scale manifold learning and dimension reduction." *IEEE Transactions on Neural Networks* 21 (10): 1576–1587.

Zhang, M., and A. A Sawchuk. 2011. "A feature selection-based framework for human activity recognition using wearable multimodal sensors." In *Proceedings of the 6th International Conference on Body Area Networks,* 92–98. ICST.

Zhang, M., and A. A. Sawchuk. 2012. "USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors." In *Proceedings of the ACM Conference on Ubiquitous Computing,* 1036–1043. ACM.

Zhang, M., and A. A Sawchuk. 2013. "Human daily activity recognition with sparse representation using wearable sensors." *Biomedical and Health Informatics, IEEE Journal of* 17 (3): 553–560.

Zhang, Sai, Cihan Tepedelenlioğlu, Mahesh K Banavar, and Andreas Spanias. 2016. "Max consensus in sensor networks: Non-linear bounded transmission and additive noise." *IEEE Sensors Journal* 16 (24): 9089–9098.

Zhang, Sai, Cihan Tepedelenlioglu, Andreas Spanias, and Mahesh Banavar. 2018. *Distributed Network Structure Estimation Using Consensus Methods.* 10:1–88. 1. Morgan & Claypool Publishers.

Zhang, Yu, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. 2016. "Highway long short-term memory rnns for distant speech recognition." In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on,* 5755–5759. IEEE.

Zhao, Feng, Lishan Qiao, Feng Shi, Pew-Thian Yap, and Dinggang Shen. 2016. "Feature fusion via hierarchical supervised local CCA for diagnosis of autism spectrum disorder." *Brain imaging and behavior:* 1–11.

Zhao, L., Q. Hu, and Y. Zhou. 2015. "Heterogeneous Features Integration via Semi-supervised Multi-modal Deep Networks." In *International Conference on Neural Information Processing,* 11–19. Springer.

Zhu, C., and W.H. Sheng. 2009. "Human daily activity recognition in robot-assisted living using multi-sensor fusion." In *Robotics and Automation, 2009. ICRA. IEEE International Conference on,* 2154–2159. IEEE.

Zhuang, Jinfeng, Ivor W Tsang, and Steven CH Hoi. 2011. "Two-Layer Multiple Kernel Learning." In *AISTATS,* 909–917.

Zien, Alexander, and Cheng Soon Ong. 2007. "Multiclass multiple kernel learning." In *Proceedings of the 24th international conference on Machine learning,* 1191–1198. ACM.