ROBUST MULTI-TASK LEARNING ALGORITHMS FOR PREDICTIVE MODELING OF SPATIAL AND TEMPORAL DATA

By

Xi Liu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2019

ABSTRACT

ROBUST MULTI-TASK LEARNING ALGORITHMS FOR PREDICTIVE MODELING OF SPATIAL AND TEMPORAL DATA

By

Xi Liu

Recent years have witnessed the significant growth of spatial and temporal data generated from various disciplines, including geophysical sciences, neuroscience, economics, criminology, and epidemiology. Such data have been extensively used to train spatial and temporal models that can make predictions either at multiple locations simultaneously or along multiple forecasting horizons (lead times). However, training an accurate prediction model in these domains can be challenging especially when there are significant noise and missing values or limited training examples available. The goal of this thesis is to develop novel multi-task learning frameworks that can exploit the spatial and/or temporal dependencies of the data to ensure robust predictions in spite of the data quality and scarcity problems.

The first framework developed in this dissertation is designed for multi-task classification of time series data. Specifically, the prediction task here is to continuously classify activities of a human subject based on the multi-modal sensor data collected in a smart home environment. As the classes exhibit strong spatial and temporal dependencies, this makes it an ideal setting for applying a multi-task learning approach. Nevertheless, since the type of sensors deployed often vary from one room (location) to another, this introduces a structured missing value problem, in which blocks of sensor data could be missing when a subject moves from one room to another. To address this challenge, a probabilistic multi-task classification framework is developed to jointly model the activity recognition tasks from all the rooms, taking into account the block-missing value problem. The framework also learns the transitional dependencies between classes to improve its overall prediction accuracy.

The second framework is developed for the multi-location time series forecasting problem. Although multi-task learning has been successfully applied to many time series forecasting appli-

cations such as climate prediction, conventional approaches aim to minimize only the point-wise residual error of their predictions instead of considering how well their models fit the overall distribution of the response variable. As a result, their predicted distribution may not fully capture the true distribution of the data. In this thesis, a novel distribution-preserving multi-task learning framework is proposed for the multi-location time series forecasting problem. The framework uses a non-parametric density estimation approach to fit the distribution of the response variable and employs an L^2-distance function to minimize the divergence between the predicted and true distributions.

The third framework proposed in this dissertation is for the multi-step-ahead (long-range) time series prediction problem with application to ensemble forecasting of sea surface temperature. Specifically, our goal is to effectively combine the forecasts generated by various numerical models at different lead times to obtain more precise predictions. Towards this end, a multi-task deep learning framework based on a hierarchical LSTM architecture is proposed to jointly model the ensemble forecasts of different models, taking into account the temporal dependencies between forecasts at different lead times. Experiments performed on 29-year sea surface temperature data from North American Multi-Model Ensemble (NAMME) demonstrate that the proposed architecture significantly outperforms standard LSTM and other MTL approaches.

Copyright by
XI LIU
2019

ACKNOWLEDGMENTS

The past more than six years of PhD life in Michigan State University has been a long and unforgettable experience for me. Back to the summer of 2012, I was an undergraduate student with poor English, basic computer science knowledge, and very unclear picture of what a researcher's daily life looks like. The past few years witnessed the unprecedented prosperous progress of multiple fields in artificial intelligence, and I felt very lucky that I could be a small member of this era of AI. When this long journey finally reaches to its finish line, I would like to acknowledge everyone who has provided me with help, support, and valuable suggestions.

First and foremost, my most sincere gratitude to my advisor Dr Pang-Ning Tan. He is quite a knowledgeable and outstanding expert in his domain, and every time when I got lost in a concept or formula, he would explain it to me very patiently no matter how trivial my it is. He is also a very good mentor in research, and would like to spend time with me discussing my research topics. When I got stuck in the research, he would enlighten me with helpful comments, inspire me with his broad knowledge, and encourage me to get it from another direction. Besides, Dr Tan would like to push me to explore new topics and techniques, and at the same time to learn new staffs, which helped me to keep up with the latest progress in my field. What is more, he has always been patient, nice, and humble. I shall never forget my first two or three years when I had limited domain knowledge and research experience, and Dr Tan helped me to build up the picture of data mining step by step. It is my honor to study and work with such a great professor and educator.

I would also like to acknowledge the help from my project collaborator Dr Lifeng Luo, who supported me with helpful domain knowledge and useful study resources. I also thank to my other PhD committee members Dr Jiayu Zhou and Dr Eric Torng, who provided invaluable comments for my dissertation.

I want to thank to my lab mates and friends Prakash Mandayam Comar, Zubin Abraham, Lei Liu, Jianpeng Xu, Shuai Yuan, Courtland VanDam, Ding Wang, Farzan Masrour, Tyler Wilson, Boyang Liu, Zheyun Feng, Lei Xu, Lanbo She, Beibei Liu, Shaohua Yang, Qi Wang, Kaixiang Lin,

Sari Saba-Sadiya, Haoyang Li, and Pouyan Hatami. It has been interesting and inspiring to talk with them. And with them, my PhD life has been joyful and a lot of fun.

Finally, I want to offer my greatest thank to my family. Without their support and love, I would not accomplish this long journey to the PhD degree. I want to thank to my beloved parents for all the unconditioned love and support they gave to me. It is their countless phone calls that warmed and encouraged me during each phase of my study. I also want to thank to my dear husband for accompanying me during each obstacle that I faced, and sharing with me all the happy things in life.

Specially, this dissertation is partially supported by the National Science Foundation under grant IIS-1615612.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Advances in data mining and machine learning have led to the development of sophisticated models for solving complex prediction tasks in various application domains, from healthcare to autonomous driving. A common strategy for solving such complex learning tasks is to decompose the overall prediction problem into smaller sub-tasks that can be solved in a more efficient and tractable manner. For example, in autonomous driving, the sub-tasks include identifying obstacles in front of the vehicle, tracking movement of other vehicles in its surrounding, recognizing street signs, and detecting lane departures. Training a global model that can be applied to all the sub-tasks will likely lead to inferior model performance due to the inherent differences among the sub-tasks. A more effective strategy would be to train a separate prediction ("local") model that can discern the relationship between the predictor and response variables for each sub-task.

Formally, let $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_T\}$ be the set of predictor variables for each of the T prediction sub-tasks, where $\mathbf{X}_i \in \mathbb{R}^{n_i \times d}$, and $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_T\}$ be the corresponding set of response variables, where $\mathbf{y}_i \in \mathbb{R}^{n_i}$. For such multi-task prediction problem, our goal is to learn a distinct model, $f_t : \mathbb{R}^d \rightarrow \mathbb{R}$, that maps the predictor variables of each sub-task t to their corresponding response value. The model for each sub-task $f_t(\mathbf{w}_t)$ is assumed to be characterized by its model parameter $\mathbf{w}_t$, which can be estimated during training by optimizing the following objective function:

$$\{w_1^*, w_2^*, \cdots, w_T^*\} = \arg\min_{\{w_t\}} \sum_t^T \mathcal{L}_t \Big[\mathbf{y}_t, f_t(\mathbf{X}_t; w_t)\Big], \tag{1.1}$$

where $\mathcal{L}_t(\cdot)$ is the loss function for sub-task t and $\{w_1^*, w_2^*, \cdots, w_T^*\}$ are the learned parameters. Each sub-task can be solved separately as the loss functions are independent of each other. This approach is also known as single-task learning (STL). In principle, STL would work well if there are sufficient amount of training data $(\mathbf{X}_t, \mathbf{y}_t)$ available for each sub-task. However, since acquiring labeled data can be expensive, the performance of STL can still be poor as its induced models are highly susceptible to overfitting when there are limited training data.

To overcome the limitation of STL, multi-task learning (MTL) approaches (MTL) have been proposed [20, 148, 147]. The model parameters for MTL are generally solved by optimizing the following joint objective function:

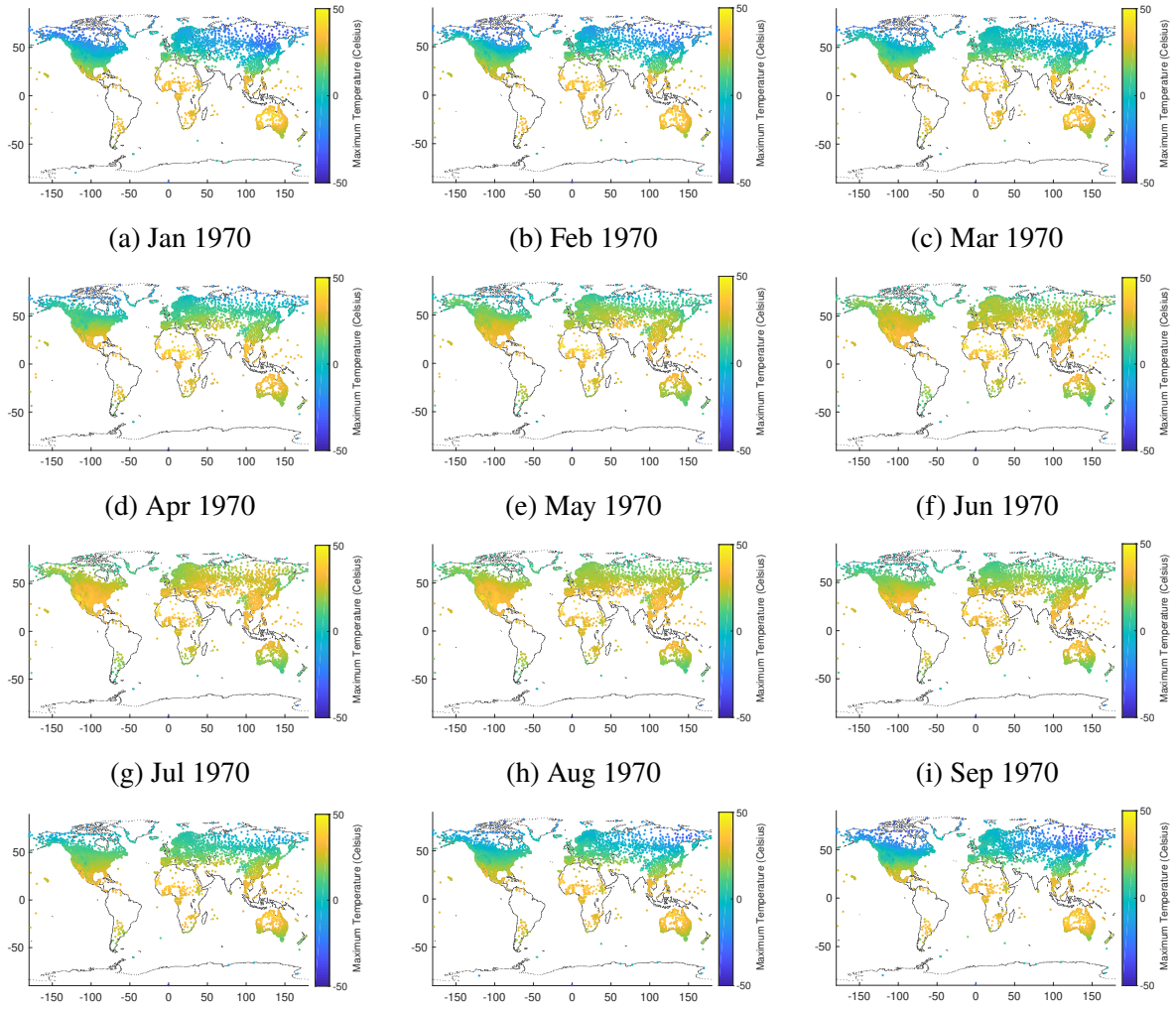
$$\{w_1^*, w_2^*, \cdots, w_T^*\} = \arg\min_{\{w_t\}} \sum_t^T \mathcal{L}_t \left[\mathbf{y}_t, f_t(\mathbf{X}_t; w_t) \right] + \Omega(w_1, ...w_t, ...w_T) \tag{1.2}$$

where $\Omega(w_1, ...w_t, ...w_T)$ is a regularization term that controls the dependencies among the parameters. The regularization enables MTL to leverage domain-specific knowledge from other related sub-tasks to prevent each model from overfitting its training data, thereby improving its generalization performance [20]. The success of MTL has been well-documented in many application domains including computer vision [47, 41], natural language processing [80, 114], and medical informatics [101, 15].

In this thesis, I will focus on the development of MTL approaches for spatial and temporal data. Specifically, several challenging problems from various application domains are investigated and novel frameworks are proposed to overcome these challenges. I will first present an overview of spatial and temporal data as well as its applications in the next two sections before summarizing the contributions of this thesis.

1.1 Spatial and Temporal Data

Spatial and temporal data are observations that contain measurements of geographic location and time information [10, 102]. Such data are pervasive across many application domains, including climate and environmental sciences [57, 85, 56, 62, 125, 96, 133, 77], neuroscience [34, 109, 131, 43, 29, 31], health sciences [83, 89, 106], social sciences [23, 51], transportation studies [71, 88], and criminology [129, 110]. One important characteristic of the data is their non-*i.i.d* (independent and identically distributed) property. The non-independence property arises due to the inherent autocorrelation of their measurements along the space and/or time dimensions. In particular, the presence of strong spatial auto-correlation implies that observations at nearby locations should be similar to each other [66] while temporal autocorrelation refers to the non-random association between a pair of observations measured at nearby times. For example, Fig. 1.1 shows the variability



(a) Jan 1970 (b) Feb 1970 (c) Mar 1970

(d) Apr 1970 (e) May 1970 (f) Jun 1970

(g) Jul 1970 (h) Aug 1970 (i) Sep 1970

(j) Oct 1970 (k) Nov 1970 (l) Dec 1970

Figure 1.1: Monthly maximum temperature observed at weather stations around the world in 1970. The data was obtained from the Global Historical Climatology Network (GHCN) [85].

of monthly maximum temperature of weather stations around the world for a one year period in 1970. Due to its high spatial and temporal autocorrelation, the monthly maximum temperature appears to change smoothly along its spatial and temporal dimensions, as illustrated in the figure. From a predictive modeling perspective, failure to account for the spatial and/or temporal autocorrelation of the response variable may lead to suboptimal local models as the predicted values may not exhibit the desired autocorrelation properties when the observations are treated independently during training. As evidenced by many previous studies, incorporating such autocorrelation into the learning framework would indeed improve model performance [39, 139, 136].

Though the non-independence property of the data suggests homogeneity of the models, its non-identically distributed property suggests there should be still notable differences in the models due to the spatial heterogeneity of the data. An example illustrating the opposing forces of spatial autocorrelation and spatial heterogeneity is shown in Fig. 1.1. Though the maximum temperature at two nearby locations are similar, there are still significant differences between the observations in the northern and southern hemispheres on a given month. From a predictive modeling perspective, the non-identically distributed property suggests that a one-size-fits-all approach using a global model to fit all the training observations is not a viable strategy as the model fails to account for the spatial differences of the observations. Instead, the modeling approach should incorporate local features that can help explain the variability observed in the data, both along the spatial and temporal dimensions.

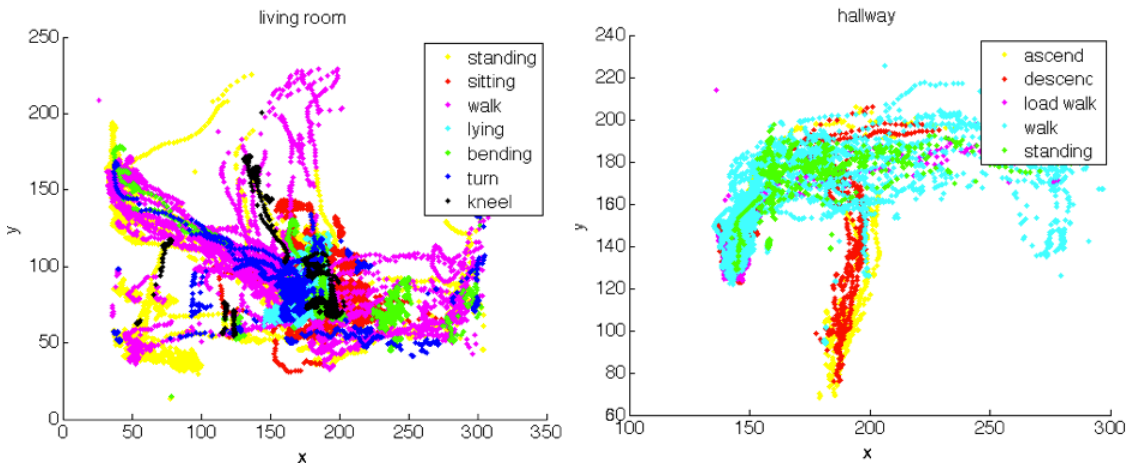
The non-*i.i.d.* property thus provides a strong motivation for using MTL for predicting modeling of spatial and temporal data. Instead of building a single (global) model, MTL addresses the non-identically distributed (heterogeneity) problem by training a separate model for each sub-task (which could be a location or a forecast lead time). MTL also alleviates the limitation of STL in terms of handling non-independent observations by allowing the models to incorporate the spatial and/or temporal autocorrelations as regularization terms in its formulation as given in Eq. (1.2).

1.2 Applications of Multi-task Learning to Spatial and Temporal Data

In this section, I will briefly describe several spatial and temporal prediction problems in which MTL can be used along with their respective applications. These applications would serve as case studies for evaluating the MTL frameworks developed in this dissertation.

1.2.1 Multi-modal Time Series Classification at Multiple Locations

The first MTL problem investigated in this thesis involves classification of multi-modal time series at multiple locations. An example application of such problem is identifying the daily activities of a human subject from the multi-modal sensor data collected in a smart home environment [122,



(a) Living Room (b) Hallway

Figure 1.2: A snapshot of the trajectories and activities of a human subject from the benchmark Sphere challenge dataset [122] (figure is best viewed in color).

76, 72, 73]. We use a benchmark user activity data from [122] for this study. Fig. 1.2 shows an example of the trajectories recorded by a tri-axial accelerometer worn by a human subject who moved around the kitchen and hallway areas of a smart home. Each colored dot represents a specific activity (walking, standing, sitting, lying down, etc.) performed by the human subject. In addition to the accelerometer sensor, RGB-D cameras and environment sensors were also installed in some of the rooms in the smart home. While the trajectory data is available everywhere from the accelerometer worn by the subject, video data from RGB-D cameras are only available in a few of the rooms (e.g., in kitchen and living room but not in bedroom nor bathroom). This introduces blocks of missing values in the video data as the subject moves from one room with RGB-D camera to another room without such camera. The modeling approach must therefore be able to account for such varying types of features available in different rooms. In addition, since the layout for each room is different, some activities are more likely to be performed in certain rooms than others. For example, the activities “ascending" or “descending" stairs are more likely to occur in in the hallway than in the kitchen or bedroom. Due to such spatial constraints, it makes more sense to develop a local model for activity recognition in each room instead of fitting a global model for all the rooms. However, due to the noisy nature of the data and the limited training samples available in some rooms, the local models are susceptible to model overfitting problem. Multi-task learning thus

provides a promising approach to address such multi-location time series classification problem.

1.2.2 Multi-location Time Series Forecasting

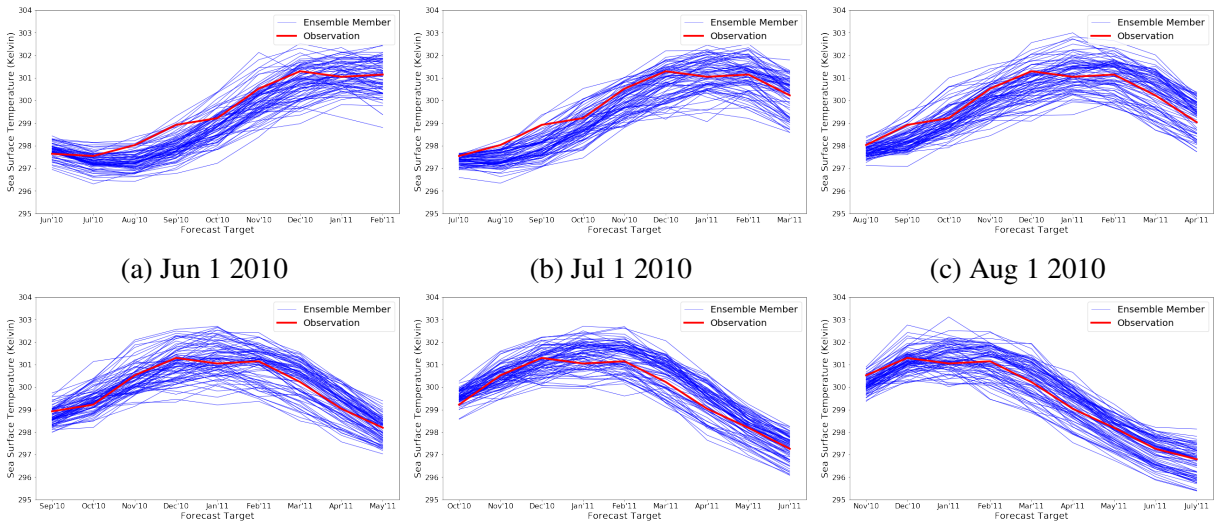
Another common prediction problem involving spatial and temporal data is multi-location time series forecasting, where each location is affiliated with a time series whose future values are to be predicted. Example applications of such problem include climate, disease incidence, and crime rate prediction. For example, in climate prediction, our objective is to predict future climate conditions at various locations based on the historical climate observations at each location as well as other auxiliary information such as local topology, vegetation, or simulated outputs from global and regional climate models. Fig. 1.1 shows an example of the monthly maximum temperature measurements in in the year 1970 for more than 70,000 weather stations. The data was obtained from the Global Historical Climatology Network (GHCN) database [85]. As previously noted, the climate data exhibit strong spatial and temporal autocorrelation, which makes it natural to apply MTL approaches to exploit such dependencies and train the prediction models at multiple locations jointly.

1.2.3 Multi-step-ahead Time Series Prediction

The third MTL problem investigated in this thesis is multi-step-ahead (i.e., long-range) time series prediction. Unlike the previous problem, which focuses only on the prediction for the next immediate future time step, the goal here is to predict the values for multiple future time steps. Each future time step is called a lead time, while the maximum lead time is known as the forecasting horizon. The multi-step-ahead time series prediction problem will be investigated in the context of its application to ensemble forecasting of sea surface temperature (SST), which is an important task due to the strong influence of ocean temperature on global climate conditions. While it is possible to apply single-step time series forecasting methods to a multi-step forecasting setting, such an approach typically requires using the predicted value of one time step to infer the value for

the next time step. This would lead to an error propagation problem, in which the error can quickly become unacceptably high even at short-range forecasting horizons.

To overcome this problem, ensemble forecasting uses a set of forecasts generated from computer-generated (physical) models to project the possible future scenarios [62]. As the computer models were developed based on the physical laws of the underlying domain, their outputs are likely to be more consistent with the true observations even for long-range forecasting horizons. For example, Fig. 1.3 shows the multi-step ahead monthly SST forecasts generated by a set of ensemble members obtained from the North American Multi-Model Ensemble (NMME) [62]. Each blue curve represents the 8-month forecasts generated by an ensemble member while the red curve represents the true SST values for the 8-month forecasting horizon. In the NMME dataset, a new set of multi-step ahead forecasts are generated by the ensemble members every month. For example, Fig. 1.3(a), shows the 8-month ahead monthly average SST forecasts generated by 80 ensemble members on Jun 1st, 2010 for the months of June 2010 until February 2011. The next set of multi-step ahead ensemble member forecasts were generated on July 1st, 2010 and are shown in Fig. 1.3(b). As the red line is encapsulated within the envelope of blue curves, the plot suggests that the ensemble members are capable of capturing the range of forecast uncertainties of SST even at 8-month forecasting horizon. Nevertheless, the ensemble member forecasts still need to be aggregated to obtain a point prediction for each lead time. This can be achieved by applying regression techniques to learn the mapping from ensemble member forecasts into point-wise prediction. However, since the skills of the ensemble members may vary from one lead time to another, it may not be wise to train only a single regression model for aggregating the ensemble member forecasts at all lead times. Instead of training a separate model for each lead time, MTL provides a promising approach for this problem by exploiting the temporal autocorrelation of the SST values at different lead times.



(a) Jun 1 2010 (b) Jul 1 2010 (c) Aug 1 2010

(d) Sep 1 2010 (e) Oct 1 2010 (f) Nov 1 2010

Figure 1.3: Sea Surface Temperature(SST) ensemble members vs. actual SST observations. From North American Multi-Model Ensemble (NMME) [62].

1.3 Research Challenges

This section presents the research challenges associated with the problems and applications described in Section 1.2.

- **MTL for Multi-modal Time Series Classification at Multiple Locations** There are several challenges that must be addressed when applying MTL to the multi-modal time series classification problem described in Section 1.2.1. First, the MTL approach must account for the temporal dependencies between the classes. For the human activity recognition application, some transitions between activities are more or less likely to occur than others. For example, a subject often "bends" before "jumping" and rarely "sits" immediately after "jumping". Incorporating such temporal dependencies into the modeling framework may potentially enhance the prediction results. Although such dependence relationship can be acquired from domain knowledge, they may not be complete nor exact enough to help recognize the activities of individual human subjects. Instead of using a pre-defined relationship, it is better to extract the temporal relationships between the classes directly from the data. In addition to the temporal dependencies, the classes may have spatial relationships as well. For

example, Fig. 1.2 shows that the classes "ascend" and "descend" stairs are more prevalent in the hallway than in the living room. It would be advantageous to have a modeling framework that can account for such spatial dependencies in the data. Finally, as previously noted, due to the varying sensor data available in different rooms, addressing the block missing value problem is another challenge that should be addressed by the modeling framework.

- **MTL for Multi-location Time-series Forecasting**

  As mentioned in Section 1.2.2, the multi-location time series forecasting problem requires building prediction models for different locations in the data. Previous research [136, 139] has mostly focused on applying MTL to jointly train the models for different locations in order to maximize their overall prediction accuracies. However, in many applications such as climate modeling, preserving the true distribution of the data is just as important as maximizing model accuracy [5] as the predicted distribution can provide useful information for planning, risk assessment, and other decision making purposes. For example, knowing the future distribution of temperature and precipitation can help climate scientists to better anticipate the severity and frequency of adverse weather events for climate impact assessment studies. In agricultural production, the predicted distribution can be used to derive statistics such as average length of future growing season or persistence of wet and dry spells, which are important metrics for farmers and agricultural researchers. However, achieving both high accuracy and preserving the distribution fit at each location is a challenge that has not been addressed by existing MTL frameworks.

- **MTL for Multi-step-ahead Time Series Prediction**

  The multi-step-ahead time series prediction problem described in Section 1.2.3 requires building a prediction model for each forecast lead time. However, the prediction error tends to grow as the lead time increases as illustrated by the SST ensemble forecasting task shown in Fig. 1.3. The plots show that the variance of the ensemble member forecasts generally increases with longer lead times. MTL can help alleviate this problem by leveraging

information from the shorter lead time tasks to regularize the predicted values for the longer lead time tasks. However, the challenge here is how to determine the relationship between the shorter and longer lead time tasks. Previous works such as [135] assume there is a predefined graph structure in terms of the task relationship between different lead times. In fact, many of the previous MTL frameworks [148] are based on some pre-defined assumption about the task relationships, e.g., using model correlation matrix [136], graph Laplacian structure [135, 40], low-rank [24, 139, 134], or model sparsity structure [8]. While these assumptions are mostly designed for general learning problems, its effectiveness for the multi-step ahead time prediction problem remains unclear. In particular, the task relationship could be nonlinear, and thus, needs to be inferred from the data. Learning the appropriate task relationship for multi-step-ahead time series prediction is a challenge to be addressed in this dissertation.

1.4 Thesis Contributions

- **Chapter 3: Multi-task Learning on Multi-modal Sensor Data for Time Series Classification**

  To address the first challenge described in Section 1.3, Chapter 3 presents a probabilistic multi-task learning framework for multi-modal time series classification. The framework learns the pair-wise temporal dependencies between the classes and incorporates such dependencies into its formulation to enhance the activity recognition performance of each classifier. It employs a softmax classifier, in which the model parameters for each class are learned jointly at multiple locations. Furthermore, to address the varying feature types at different locations, the framework decomposes its feature set into two parts–a common feature set (for all locations) and a location-specific feature set. While the parameter values for the common feature set are learned jointly across all locations, the location-specific ones are learned independently for each location. This strategy enables the proposed MTL framework to address the block-missing value problem.

- **Chapter 4: Distribution Preserving Multi-Task Regression for Multi-location Time Series Forecasting**

  To address the challenge described in the previous section for multi-location time series forecasting, Chapter 4 presents a novel distribution preserving MTL framework for spatio-temporal data. The proposed framework is unique in that it integrates both distribution and point-wise data fitting in a unified learning formulation. A non-parametric kernel density estimation approach is employed to fit the marginal distribution of response variable along with an L^2-distance measure used to estimate the divergence between the predicted and true distributions. Parameter sharing between models trained at different locations is enforced through their low-rank structure along with a graph laplacian regularizer based on Haversine distance between locations. The effectiveness of the proposed approach is then demonstrated through extensive experiments using a 45-year precipitation dataset for more than 1500 weather stations in the United States.

- **Chapter 5: Multi-task Hierarchical LSTM Structure for Multi-step-ahead Time Series Prediction**

  To address the third challenge described in the previous section, Chapter 5 presents a multi-task deep learning architecture for the multi-step-ahead ensemble forecasting problem. The architecture considers the prediction for each lead time as a separate learning task and employs a novel two-layer hierarchical LSTM structure to learn a nonlinear relationship between the tasks. The first LSTM layer of the hierarchy learns a latent representation for each lead time task, taking into account their temporal dependencies. This enables the proposed framework to leverage information from shorter-term lead time tasks to improve the prediction for longer-term tasks. The second layer of the hierarchy learns a feature representation for the generation times of the forecasts. Specifically, given a specific lead time, it assumes that the hidden representations of the forecast generation times are related in a sequential way using an LSTM. This allows the architecture to capture the temporal autocorrelation between

forecasts generated at different times. The entire architecture is trained in an end-to-end fashion and evaluated on an ensemble of monthly sea surface temperature data for a large study region in the Pacific ocean.

1.5 Other Research Contributions

In addition to my research contributions to the development of MTL frameworks for spatial and temporal data in this thesis, I have also developed a multi-label learning framework that incorporates taxonomy-type relations for categorizing mobile apps. With the proliferation of smart devices and apps markets, there is a pressing need to develop automated techniques for apps categorization. While existing app markets such as Google Play and Apple Store do provide their own list of app categories, there are several issues with the existing market categorizations.

1. Limited granularity. With the explosive growth of mobile apps has resulted in huge amount of apps per category, rendering the task of searching in a category to be laborious and time consuming. The granularity of current mobile app categorization is also too coarse to effectively distinguish between apps assigned to the same category.

2. Lack of objectivity. The mobile apps in online app stores are typically classified manually based on the subjective judgment and may not agree with the actual use of the apps.

3. Limited expressiveness. As pointed out in [70], the hard, exclusive labeling results in a large number of multi-category apps missing from their appropriate categories. For example, the Instagram app [2] is found only in the social networking category but not in the photography category, even though it has functionalities related to both categories.

To overcome these limitations, I propose a novel approach in in [74] to automatically label apps with a richer and more flexible categorization. In order to label apps with a finer-grained ontology than the original categories the app market provides, a detailed categorization is leveraged from an application domain that closely resembles that of mobile apps (*e.g.*, Google Ad Preference ontol-

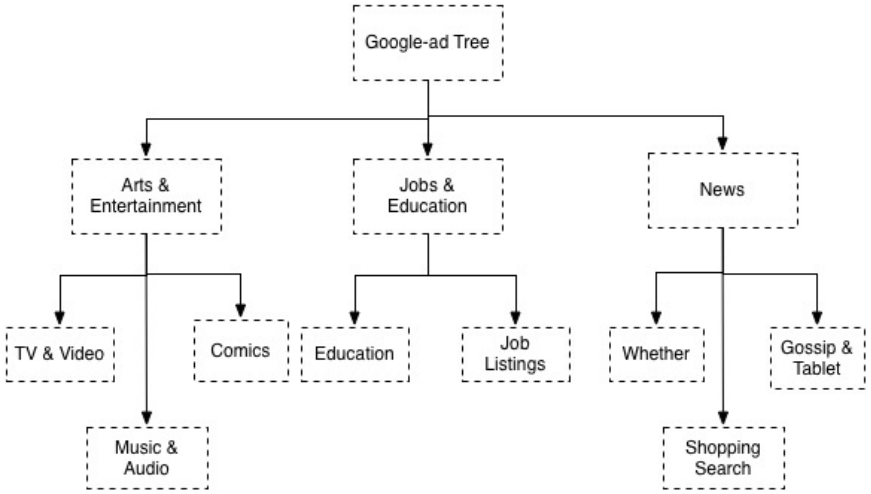


Figure 1.4: A snippet of a custom input categorization

ogy [1]). Unlike the flat structure of the original market categorization, the target categorization structure is hierarchical illustrated in Fig. 1.4.

The proposed framework uses a semi-supervised Non-negative Matrix Factorization (NMF) approach to classify the apps. The framework takes into account various factors, such as the class labels associated with sampled training set apps, feature vectors of unlabeled apps, as well as the side information encoding affinity relationship between input categories. All of this information are integrated into a unified learning framework that automatically generates (i) the predicted labels for previously uncategorized apps, (ii) the important features characterizing different classes, and (iii) a modified inter-class similarity matrix that better fits to specific characteristics of mobile apps. The proposed semi-supervised NMF framework is designed to minimize the following objective function:

$$\begin{aligned} \min_{\mathbf{Y}_u,\mathbf{L},\mathbf{B}} \quad & ||\mathbf{Y}_l - \mathbf{X}_l\mathbf{W}||_F^2 + ||\mathbf{Y}_u - \mathbf{X}_u\mathbf{W}||_F^2 + \beta||\mathbf{L}||_F^2 \\ & + \gamma||\mathbf{B} - \mathbf{P}||_F^2 \\ \text{s.t.} \quad & \mathbf{W} = \mathbf{L}\mathbf{B},\ \mathbf{B} \geq 0, \mathbf{L} \geq 0, \mathbf{Y}_u \geq 0 \end{aligned} \tag{1.3}$$

where the subscripts l and u denote the labeled and unlabeled data sets; n_l and n_u denote the number of labeled and unlabeled examples, respectively. $\mathbf{Y}_l \in \mathfrak{R}_+^{n_l \times k}$ and $\mathbf{Y}_u \in \mathfrak{R}_+^{n_u \times k}$ are the class indicator matrices, where k is the number of classes. $\mathbf{X}_l \in \mathfrak{R}_+^{n_l \times d}$ and $\mathbf{X}_u \in \mathfrak{R}_+^{n_u \times d}$ are the

feature matrices associated with the labeled and unlabeled data, respectively. To help guide the app categorization with custom ground truth categories, the NMF framework can be modified to utilize side information about the classes in the input categories. Specifically, the side information can be represented as a $k \times k$ class similarity matrix $\mathbf{P}$. Each entry $\mathbf{P}_{ij}$ denotes the similarity between classes i and j, which is obtained by checking their sibling relationship (i.e, whether i and j share a common parent in Google Ad Tree). Formally,

$$\mathbf{P}_{ij} = \begin{cases} p = \frac{1}{\text{\# of levels}} & \text{if } i \text{ and } j \text{ are siblings,} \\ 0 & \text{otherwise.} \end{cases} \tag{1.4}$$

The performance of the proposed framework was evaluation on $1,065$ apps from Google Play. Each app is characterized by a tfidf feature vector of length 7,745. The classification performance for each class is measured by Area Under Curve (AUC) of Receiver Operating Characteristic (ROC) [119]. Using right y-axis of Figure 1.5, a histogram of 49 Google-ad categories that represents the number of apps in each category (in black) is plotted. Using left y-axis of Figure 1.5, two curves that represent AUCs of proposed approach for each class (in red), and AUC of logistic regression (in blue) are plotted. For large classes with more than 50 apps, both the proposed approach and logistic regression yield similarly good AUC performances. However, for smaller classes with less than 50 apps, the semi-supervised NMF clearly outperforms logistic regression.

1.6 Related Publications

Some chapters in this dissertation are based partially on the following publications: Chapter 3 is based on three papers entitled "STARS: Soft Multi-Task Learning for Activity Recognition from Multi-Modal Sensor Data" [76], "Human daily activity recognition for healthcare using wearable and visual sensing data" [72], and "Location-based Hierarchical Approach for Activity Recognition with Multi-modal Sensor Data" [73]. Chapter 4 is from "Distribution Preserving Multi-Task Regression for Spatio-Temporal Data" [75], which appeared in the Proceedings of the 2018 IEEE International Conference on Data Mining. Chapter 5 is based on the materials from a paper that is currently under review for the 2019 ACM SIGKDD International Conference on

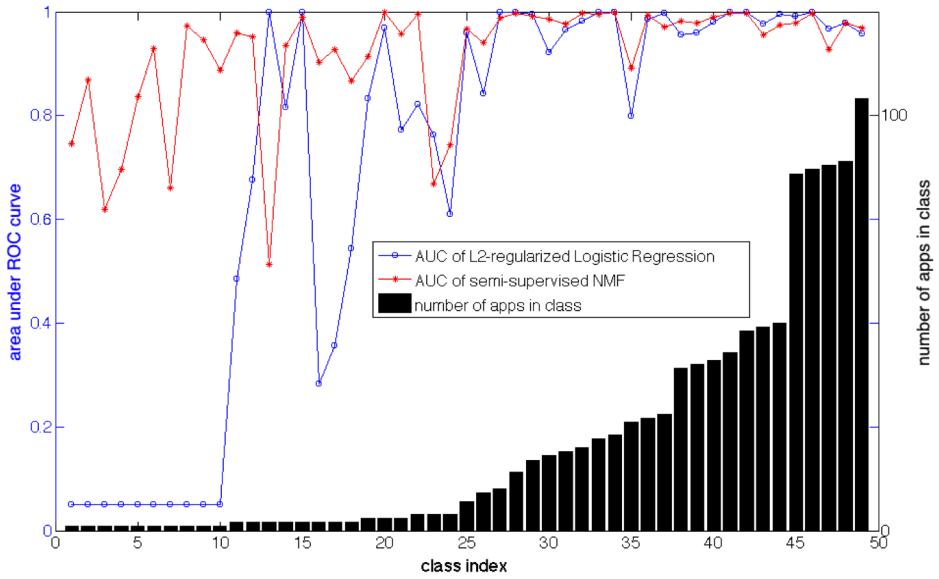


Figure 1.5: AUC comparison for classes with different number of apps.

Knowledge Discovery and Data Mining. My other publications include "Macro-scale mobile app market analysis using customized hierarchical categorization" [74], "MUSCAT: Multi-Scale Spatio-Temporal Learning with Application to Climate Modeling" [134], and "WISDOM: Weighted incremental spatio-temporal multi-task learning via tensor decomposition" [139].

1.7 Thesis Outline

The rest of this thesis is organized as the following. Chapter 2 presents the background and previous literature related to my thesis topic. Chapter 3 describes my research on multi-modal time series classification for human activity recognition while Chapter 4 introduces the distribution preserving regression framework for multi-location climate forecasting. Chapter 5 presents the MTL framework for multi-step-ahead ensemble forecasting of sea surface temperature. Finally, conclusions and future research directions are discussed in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

2.1 Spatial and Temporal Data Mining

Spatial and temporal data refers to observations containing variables that vary over space and time [10, 102]. Due to the non-*i.i.d* property of spatial and temporal data, novel spatial and temporal data mining techniques have been developed to discover interesting patterns and models from such data [107], which include:

- Spatio-temporal outliers [116, 108], which correspond to observations whose non-spatial-temporal features are significantly different from the rest of the data set. An example application of this approach is to predict extreme climate events from precipitation data [132].
- Spatio-temporal couplings [42, 90], which correspond to multiple events that occur in close spatial or temporal proximity; For example, studying the spatio-temporal coupling of traffics helps better planning on the road.
- Spatio-temporal partitioning or clustering [63, 104], which correspond to groups of observations that are similar to each other along the space or temporal dimensions. Spatio-temporal hotspots is a special topic of clustering, where high intensity of instances is observed to happen in a certain region or time period;
- Spatio-temporal change footprints [150], which correspond to changes in the data distribution or underlying model over space or time; For example, in industrial process, the change of statistical index of sensors may indicate system fault; the raise of reported infections may indicate the epidemic outbreak.
- Spatial and temporal prediction, which corresponds to the task of inferring the response values either at previously unknown locations or for future time steps. Example applications

include predicting crime rate or future climate condition at a given location.

This thesis will mainly focus on the spatial and temporal prediction problems. Conventional predictive modeling techniques (e.g. logistic regression, ridge regression, decision tree and so on) usually assumes the data to follow *i.i.d* property, which are not suitable for spatial and temporal data. Therefore, new techniques are developed to build connections between consecutive locations or time steps. Some examples are summarized as the following categories:

- Incorporating temporal dependencies and predicting on time. Techniques like autoregressive integrated moving average (ARIMA) [81] are specifically designed for time series forecasts; besides, some approaches designed for sequential data are also demonstrated to be efficient on predicting on time series, such as conditional random field [60] and recurrent neural network [100].

- Incorporating spatial dependencies and predicting on space. Techniques like spatial autoregressive regression (SAR) [58] and kriging [91] are developed to predict for unknowns over space;

In sum, on the one hand, due to the non-independence property, it is unwise to build an independent local model for each spatial/temporal unit, as what Tobler's first law of geography says: "*Everything is related to everything else, but near things are more related than distant things*" [121]; on the other hand, due to the non-identical property, it is unreasonable to build a single global model and expect it to fit for all spatial/temporal units. Since the local model ignores the generalization while global model lacks uniqueness, it is suggested to develop new approaches that can incorporate data dependencies over space/time and at the same time keeps distinct spatial/temporal attributes.

2.2 Multi-task Learning

Multi-task learning (MTL) is a machine learning technique which jointly deals with multiple related modeling tasks by incorporating the correlations/dependencies among tasks [20, 148]. This

approach is useful as many complex prediction problems can be decomposed into multiple learning tasks. Building a single global model for all tasks may not be effective because the global model is too general to fit each task well. In contrast, building a local model independently for each task would require having a sufficiently large training set for each task to avoid model overfitting problem. Multi-task learning addresses this problem by leveraging domain-specific information to enable the pooling of information across different tasks in order to improve its model generalization [105].

MTL is employed under two scenarios: 1). predictive modeling for multiple homogeneous objects; 2). learning with heterogeneous auxiliary tasks. [105, 69]. The latter one appears in many studies in computer vision [35, 146, 38] and natural language processing [27, 84, 11], where heterogeneous tasks are processed. For example, in [146], the facial landmark detection is enhanced with the help of correlated auxiliary tasks such as gender classification, head pose classification, age estimation, facial expression recognition. In [27], a weight-sharing structure is built to jointly learn several language processing predictions. In this thesis, I mainly focus on the former case, and a few existing MTL structures are reviewed in the following.

2.2.1 MTL Based on Encoding Graph Structures

In many applications, the pair-wise relations between N tasks can be pre-defined with $N \times N$ similarity matrix. Different assumptions are made to quantify the relatedness between tasks. For example, in multi-location geospatial predictions, it is reasonable to assume that locations that are geographically closer or demographically similar share more similarities in models than those far away from each other [99].

One of the popular way to embed the tasks similarities into an objective function is to use graph Laplacian regularizer. Graph Laplacian is commonly used in spectral clustering [127] to optimize graph partitioning. The spectral property of a graph is examined with the eigenvectors of Laplacian matrix, with the property that Laplacian matrix is positive semidefinite [119]. This property also makes Laplacian matrix a good way for smoothness. For example, in [94], the graph regularizer is used for image denoising by assuming the pixel patches are smooth with respect to a pre-defined

graph.

The semidefinite property of Laplacian matrix is borrowed by MTL approaches to build connections between model parameters from different learning tasks. Treating the model of each task as a single node in the graph, and embedding the pair-wise similarities $\mathbf{A} \in \mathfrak{R}^{N \times N}$ as weighted edges, the graph Laplacian minimizes the weighted Euclidean distance of linear coefficients $\mathbf{w}_i$ from different models [148], in order to make a pair of individual learning tasks with higher proximity to have closer coefficients, and vice versa.

$$\begin{aligned} \min_{\mathbf{W}} \quad & \mathcal{L}(\mathbf{W}) + \lambda Tr[\mathbf{W}^T \mathbf{L} \mathbf{W}] \\ = \quad & \mathcal{L}(\mathbf{W}) + \frac{\lambda}{2} \sum_{i,j}^{N} A_{i,j} ||\mathbf{w}_i - \mathbf{w}_j||_2^2 \\ \text{s.t.} \quad & \mathbf{L} = \mathbf{D} - \mathbf{A}, \\ & \mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ...\mathbf{w}_N] \end{aligned}$$

Laplacian matrix can be generated with both graph structure matrix and similarity matrix. In [148, 68], the Laplacian matrix is generated directly from the topology of graph. In [40], a structure matrix is defined to make each model approaching the mean model. In [149, 135], each time point in the time series is treated as a task, and models of neighboring tasks are built to be smooth. In [136], each location is treated as a task, and the inverse of a modified variogram is used to measure similarities between tasks.

2.2.2 MTL with Low Dimensional Subspace

Another assumption made for relations among tasks is that individual models share a low dimensional subspace. Related works can be categorized in two ways. The first way uses traces norm as regularizer to minimize the common rank of linear coefficients [55, 9] from different learning tasks. Many variations of this approach emerges based on its basic formulation [24, 25].

$$\begin{aligned} \min_{\mathbf{W}} \quad & \mathcal{L}(\mathbf{W}) + \lambda ||\mathbf{W}||_* \\ \text{s.t.} \quad & \mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ...\mathbf{w}_N] \end{aligned}$$

The second way employs matrix factorization. It assumes that linear models $\mathbf{w}_i$, $i = 1, 2, ...N$ from different learning tasks share a set of base models, and each individual model is a linear combination of these base models [138, 136] as the following.

$$\begin{aligned} \min_{\mathbf{W}} \quad & \mathcal{L}(\mathbf{W}) + \lambda_1 \Omega_1(\mathbf{U}) + \lambda_2 \Omega_2(\mathbf{V}) \\ \text{s.t.} \quad & \mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ...\mathbf{w}_N] \\ & \mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ...\mathbf{v}_N] \\ & \mathbf{W} = \mathbf{U}^T\mathbf{V} \text{ or } \mathbf{w}_i = \mathbf{U}^T\mathbf{v}_i \end{aligned}$$

where $\mathbf{w}_i \in \Re^{d\times 1}$ is the linear parameter vector for individual task i. $\mathbf{U} \in \Re^{d\times k}$ represents the k shared base models, and $\mathbf{v}_i \in \Re^{k\times 1}$ describes the coefficients for linear combination of the base models. Since $d > k$, the matrix decomposition of $\mathbf{W}$ makes models from different learning tasks to share a common low rank space.

2.2.3 MTL with Incomplete Multi-source Data

A special case of MTL comes when the predictors of different tasks come from both common sources and different sources. A typical example is multi-modal sensor data [122, 145] when different sensors are not available all the time, leading to block-missing problem. It is unwise to discard the incomplete data instances or interpolating missing values. In [145], data is partitioned into multiple groups according to the data source available for each data instance. An individual model is built for each group of data. And it assumes that the model parameters corresponding to a certain source are learned jointly and have common sparsity with $L_{2,1}$ norm.

2.2.4 MTL in Deep Learning

Two most popular multi-task learning strategies in neural networks are hard parameter sharing and soft parameter sharing of hidden layers [105]. For hard parameter sharing case, the parameters of a number of layers are shared by different learning tasks, while the rest layers are task-specific [21]. An illustration is shown in Fig 2.1. For example, in the speech recognition problem in [53],

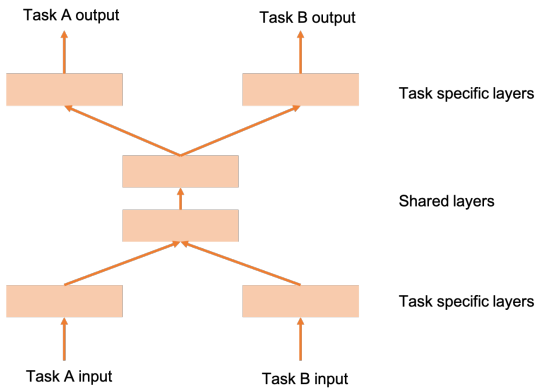


Figure 2.1: Hard parameter sharing or layer transfer in multi-task DNN.

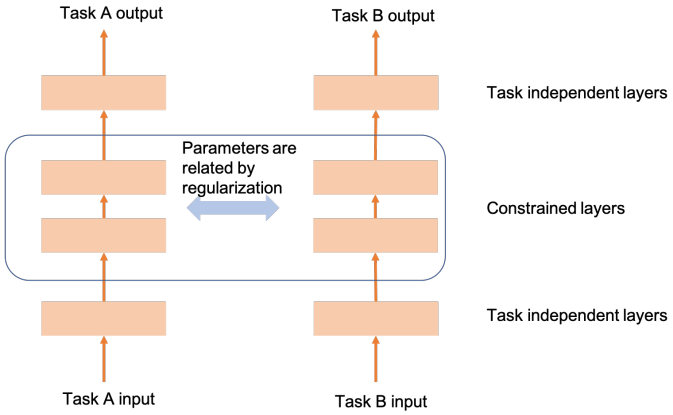


Figure 2.2: Soft parameter sharing or conservative training in multi-task DNN.

the recognition with each language is treated as a learning task, and DNN-typed frameworks are developed. It assumes that learning tasks from different languages share common hidden layers and only distinguish with each other on the softmax layer. It has been discussed in many previous literatures how shared parameters help DNN to avoid overfitting. It is a case-to-case decision to choose which layers should be shared. For example, in speech recognition, the last few layers are usually shared because it is believed that the last few layers are not dependent on the speakers. However, in image recognition, the first few layers are shared because they are capable to capture the most basic and commonly seen patterns like lines and curves [4]. Comparatively, the soft parameter sharing or conservative training illustrated in Fig 2.2. It is inspired by regularizer skills from traditional MTL approaches, and constrains the parameters learned from different tasks to be related [37, 143].

2.2.5 MTL for Multi-label Learning

Multi-label learning is a popular topic in the data mining field. Instead of labeling each data instance with only one category, multi-label learning gives multiple categories. Each labeling task results in an individual binary classifier to determine whether a data instance has this label or not. One straight forward solution is to learn for each labeling task independently. However, in many cases, there are insufficient training data for each label due to the expensive and time-consuming manual labeling process. And label-independent learning in this case will most likely result in serious overfitting problem [79]. Therefore, it is beneficial to extend the applications of MTL to multi-label learning. Instead of treating each labeling task as an independent binary classification problem and ignoring the interdependencies between labels, individual labeling problems are taken as related learning tasks [33]. Multi-label learning is considered to have many common parts with MTL, since it is a type of structured output prediction [32] which shares the hypothesis space among models [113]. The idea of MTL has been employed to solve multi-lable learning in many applications, like multi-label image classification [79, 54] and face attributes recognition [41].

There are various types of relations among labels. For example, in multi-label activity recognition, different activities are sequentially dependent. It is essential to quantify these relations by learning from data and help the recognition performance [76]. Another very commonly seen dependence label structure is taxonomy, like Google Ad ontology [1]. Doing single-task learning in this case definitely ignores the taxonomy structure and fails when the label distribution is very imbalanced. To do MTL for multi-label learning with pre-defined label taxonomy, the representation of the taxonomy needs to be formulated and incorporated into the model. However, this approach fails when the pre-defined taxonomy is not perfectly designed to serve to the data. For example, in the mobile apps categorization problem in [74], while the existing flat-structured app market categorization is limited by its lack of granularity and expressiveness, other finer-grained hierarchical categorization chosen from resembling domains still cannot guarantee to cover all concepts and aspects of mobile apps. To address the challenges, in [74], a customized hierarchical multi-task learning framework is built for multi-label categorization. The framework takes the

mobile apps categorization as an example. It leverages a fine-grained taxonomy as a guide, labels the mobile apps with finer categories, and at the same time induces a customized taxonomy.

2.3 Summary

In this chapter, the non-*i.i.d* property of spatial temporal data is discussed, and the motivation of employing multi-task learning on spatial and temporal data mining is presented. Then, several state-of-art MTL approaches are surveyed.

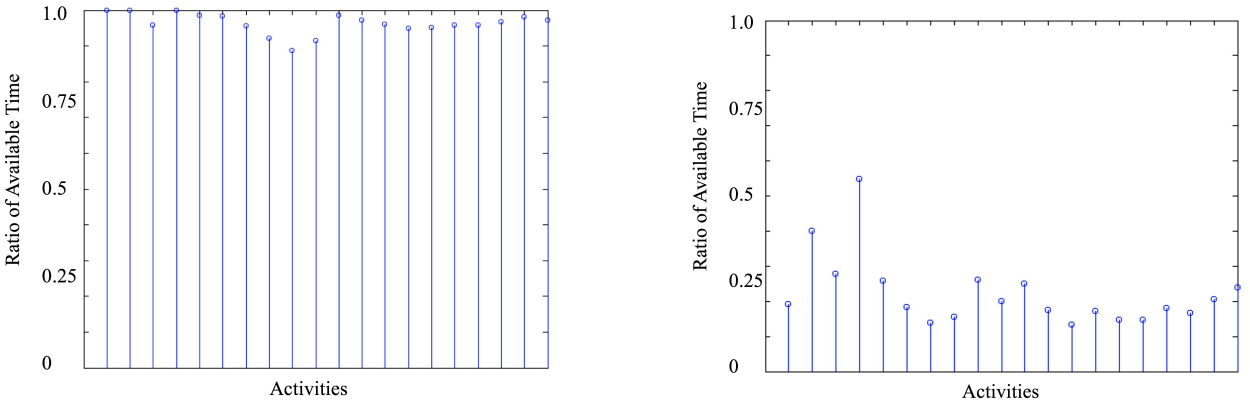
However, these existing approaches are still not enough to address all challenges on spatial and temporal data as mentioned in Section 1.3. In this thesis, I am going to investigate into some challenging problems specifically in the domain of spatial and temporal data, and develop frameworks for each targeting problem.

CHAPTER 3

SOFT MULTI-TASK CLASSIFICATION FOR ACTIVITY RECOGNITION FROM MULTI-MODAL SENSOR DATA

Rapid advances in the development of inexpensive, low-power, wireless sensing technology have enabled the deployment of sensors ubiquitously in a smart home environment to support various applications, from personal safety and security to water conservation and energy management. Real-time data generated from the myriad of sensors in the smart home provide a unique opportunity for monitoring daily living activities, alerting the residents or the authorities if any unusual activities are detected. The ability to accurately recognize human activities from the multi-modal sensor data is essential to support such applications.

However, classifying human activities from smart home sensor data is not a trivial task for several reasons. First, the sensor data are often noisy, and thus, require substantial preprocessing to extract discriminative features for the classification task. Second, the data are heterogeneous and may vary depending on the type of sensors deployed for monitoring the user activities. For example, wearable sensors such as accelerometers would generate data continuously at all times unlike other sensors such as motion detectors and surveillance cameras, which may only be available in certain rooms. For example, Fig.3.1 shows the percentage of time in which data from two sensors—accelerometer and surveillance camera—are available for each human activity in the smart home dataset investigated in this study. The results suggest that the accelerometer data is available at all times for most of the classes (human activities) whereas the surveillance camera data has a more imbalanced and irregular distribution as they are affected by the user's location as well as the rooms where the cameras are deployed. Thus, one of the key challenges is to develop a modeling approach that can handle the multi-modal sensor data, whose availability varies from one location to another depending on the sensor placement. Furthermore, the modeling approach must consider the imbalanced class distribution in different rooms since some activities could be restricted to certain locations only (e.g., one will more likely lie down in a bedroom or living room than in a



(a) Accelerometer.

(b) RGB-D camera.

Figure 3.1: Percentage of time data from an accelerometer and RGB-D camera are available for each human activity. The list of activities are shown in Table 3.1.

kitchen).

The activities performed by each user can be represented by a sequence of actions, where the transition from one action to another proceeds in a continuous fashion. Since the data are collected and annotated at discrete time periods, some activities could be interleaved together in the same time period (e.g., walking and turning at the same time or going from a standing posture to a bending and eventually kneeling position). For example, Fig 3.2 shows a 30-second segment of user activity from the labeled data used in this study. Since there could be more than one activity performed in each second, each class label (human activity) is associated with a confidence score, represented by its gray scale color. One of the goals of this study is to develop a modeling approach that can leverage the soft labels to determine the probability an activity is performed at a given time period. The temporal dependency between activities is another factor that must be taken into consideration. For example, the `lie-to-sit` transition activity typically occurs between the `lie` and `sit` postures. However, we do not expect the sequences to contain transitions from `lie` to `jump` activities. How to effectively quantify these temporal dependencies and incorporate into the modeling framework is another challenge that needs to be addressed. Although such constraints can be pre-defined from domain knowledge, they may vary depending on the dataset used. Instead of encoding them as hard constraints, my goal is to infer the temporal dependencies automatically

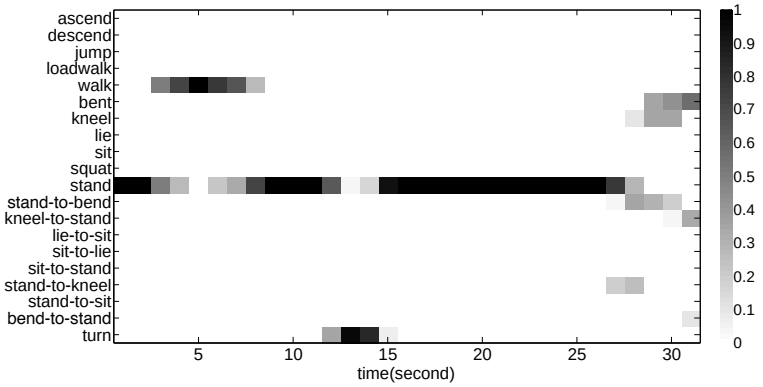


Figure 3.2: A segment of ground truth activities.

from the data.

To address these challenges, this thesis presents a soft classification approach for activity recognition in a smart home environment. The approach employs a softmax classifier to predict user activities in a sequence based on the multi-modal sensor data available. Training a global softmax classifier is not effective since some features (e.g., surveillance camera data) are only available in certain rooms. Imputing their missing values may introduce errors into the model while discarding the data with incomplete features may lead to suboptimal models. Conversely, training a local model for each room is also not the answer due to the limited training data available for some rooms and the large number of classes involved. To overcome this limitation, the proposed approach allows the local models for all the rooms to be jointly trained, and takes into account the relationship between the local models and the varying types of features available. Specifically, the framework enables the model for predicting, say, the `walk` activity in one room, to be related to the same activity in another room even if their features are not identical. This is accomplished by decomposing the weight matrix associated with the prediction of each class into a set of low rank latent factors, where the decomposition is performed only on the common features for all rooms. Using a real-world multi-modal sensor dataset [122] as the case study, I showed that the proposed framework is more effective than other sequential and non-sequential classification algorithms, including multinomial logistic regression and conditional random fields.

Table 3.1: List of human activity classes from the Sphere challenge data [122].

ascend	bent	stand	sit-to-stand
descend	kneel	stand-to-bend	stand-to-kneel
jump	lie	kneel-to-stand	stand-to-sit
loadwalk	sit	lie-to-sit	bend-to-stand
walk	squat	sit-to-lie	turn

3.1 Preliminaries

Consider a multi-modal sensor dataset, $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_R\}$, where each $\mathcal{D}_r = (\mathbf{X}^r, \mathbf{Y}^r)$ is the training set for room r. Furthermore, each $\mathbf{X}^r = \mathbb{R}^{N_r \times d_r}$ corresponds to the data matrix derived for room r, where N_r is the number of training examples available and d_r is the number of features. For notational convenience, we denote $\mathbf{X}_{i:}$ as the i-th row of matrix $\mathbf{X}$ and $\mathbf{X}_{:j}$ as its j-th column. The sensor data considered in this study [122] include 3-d acceleration features generated by a portable triaxial accelerometer worn by the subject, RGB-D camera data, and location data from passive infrared (PIR) and strength of acceleration signals (RSSI) recorded by access points location in different rooms. The raw sensor data are preprocessed to extract various features (e.g., kurtosis, frequency, and entropy of accelerometer time series and bounding box information about subjects from RGB-D camera data) associated with the human activities measured at every 1 second interval. I apply the feature extraction and preprocessing methods as described in one of my previous publication in [72].

Let $\mathbf{Y}^r \in [0, 1]^{N_r \times K}$ be the class membership matrix for all N_r observations in room r, where $\mathbf{Y}^r_{ik} \in [0, 1]$ denotes the confidence score for the i-th training instance in room r belonging to the k-th class. There are altogether 20 classes in this dataset, which are divided into 3 groups: (1) Active motion (a), which include activities such as ascending or descending stairs, jumping, and walking, (2) Stationary postures (p), which include bending, sitting down, and standing, and (3) Transition movements (t), which include stand-to-bend, lie-to-sit, sit-to-lie, and stand-to-kneel. The complete list of classes is shown in Table 3.1.

3.1.1 Sensory Data Used for Human Activity Recognition

I investigated the feasibility of using data generated from the following four sources of sensoring measurements. All of these sensors are cheap and widely embedded sensors in current smart phone, fitness band or motion sensing input devices, such as Xbox 360 and Asus Xtion Pro.

- **Acceleration**: A portable triaxial accelerometer on the wrist of a subject to record the real-time acceleration in all three spatial dimensions (X-Y-Z) in real time.
- **Bounding box from an RGB-D camera**: which combine RGB color information with per-pixel depth information. It is easy to capture and extract the moving subject with a bounding box from the raw image frames with various libraries and SDKs, such as OpenCV [92], OpenNI [93], point cloud library (PCL) [98] and Microsoft Kinect SDK [86].
- **PIR and RSSI**: Localization sensors which estimates the subject's locations in real time, in either a passive way, or an active way. While the passive sensors tell the appearance of a subject by detecting light, radiation from the human bodies (e.g.passive infrared sensor), the active sensors measures the location of a subject by sending out signals (e.g.Received Signal Strength Indication).

In this chapter, I will illustrate my preprocessing and proposed method mainly by taking the example of the data provided in "*SPHERE Challenge: Activity Recognition with Multimodal Sensor Data*" [122] for indoor human activity recognition (denoted as "Sphere data" in the rest of the text).

3.1.2 Feature Extraction

We target to predict human activities within each second. Thus, I firstly divide the entire time series into 1-second-length segments. Based on each time segment, I extract useful features to discriminate activities.

- **Features from Acceleration**

- Kurtosis [12]: Describes the tailedness of a segment of time-series acceleration data.
- Approximate Entropy [97]: Describes the unpredictability of fluctuations over a segment of time-series acceleration data.
- Top-10 Frequency by FFT [14]: Distribution of the resultant time-series energy in frequency domain.
- FFT distribution kurtosis: Describes tailedness of the resultant energy distribution in frequency domain.
- Average Jerk [130]: Average rate of change of acceleration on each axis.
- Average Absolute Value: Average absolute acceleration on each axis
- Average Value [14]: Average acceleration on each axis.
- Median: Meidian acceleration on each axis.
- Standard Deviation [14]: Standard deviation of acceleration on each axis.
- Maximum Value: Maximum acceleration on each axis.
- Minimum Value: Minimum acceleration on each axis.
- Maximum Absolute Value: Maximum absolute acceleration on each axis.

All the above features are selected because of their capability to distinguish between human activities. For example, during exploring the Sphere data, it can be found that class *"a_jump"* gets much higher maximum acceleration than any other activities (Fig 3.3). For another instance, while most of the activities don't repeat periodically, there are still some activities which show obvious repetitive patterns(e.g. *"a_ascend"*, *"a_walk"*). With Fast Fourier Transform(FFT), it can be observed that their distributions of energy in frequency domain (e.g. Fig 3.4a and Fig 3.4b) are apparently different from those who do not have periodic patterns (e.g. Fig 3.4c and Fig 3.4d). Most of these repetitive activities get relatively higher energy in low-frequency bands, while others get comparable energy in all frequency bands.

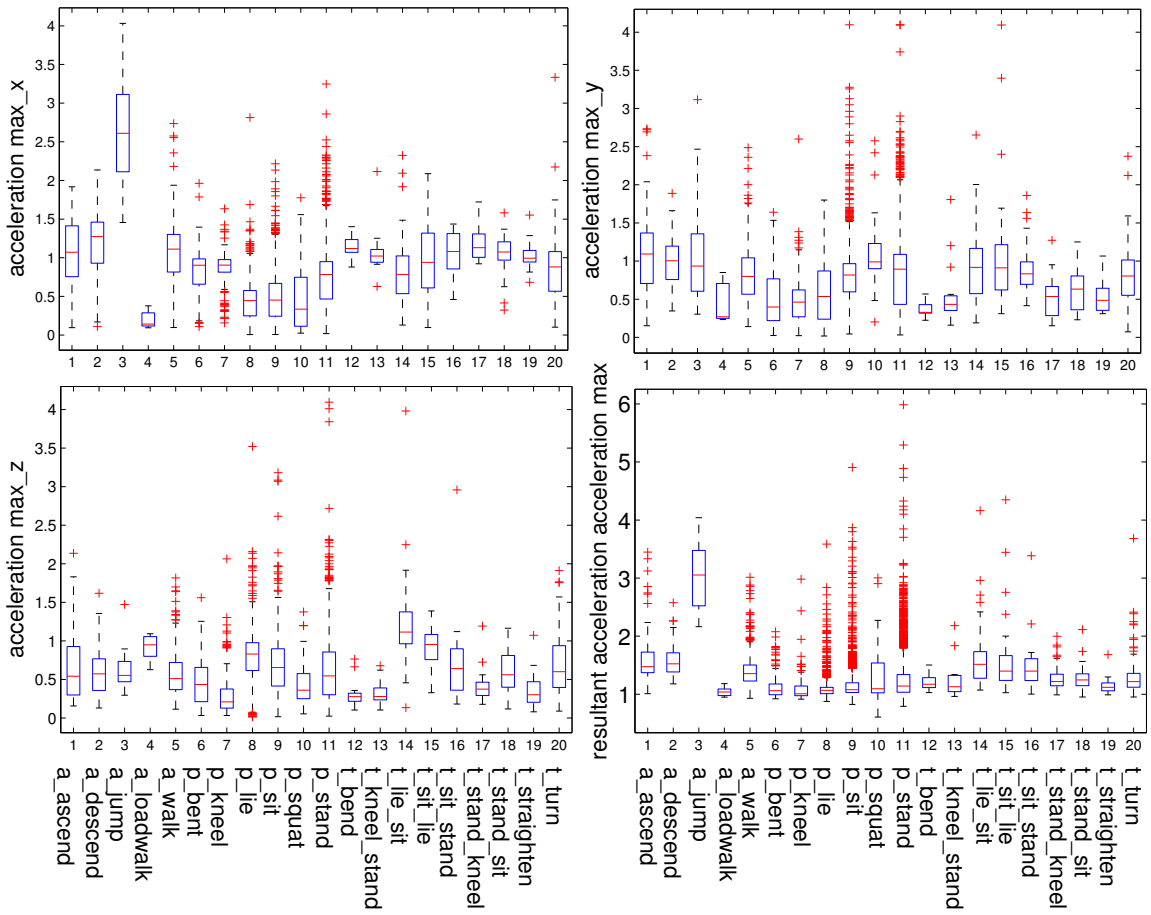
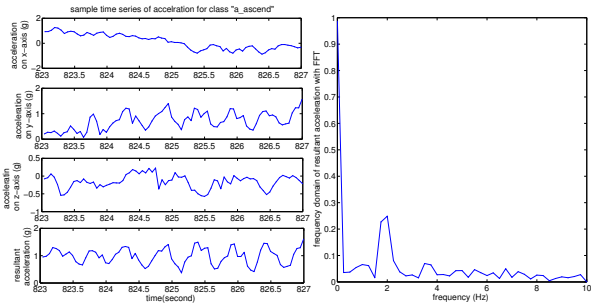
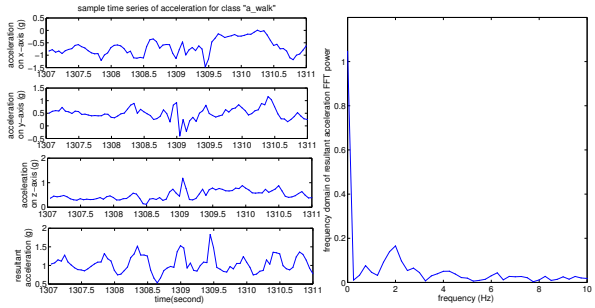


Figure 3.3: Distribution of the maximum acceleration for each activity class.(The line in the middle of each box is the sample median; The tops and bottoms of each "box" are the 25th and 75th percentiles of the samples, respectively; The whiskers are lines extending above and below each box; Observations beyond the whisker length are marked as outliers)

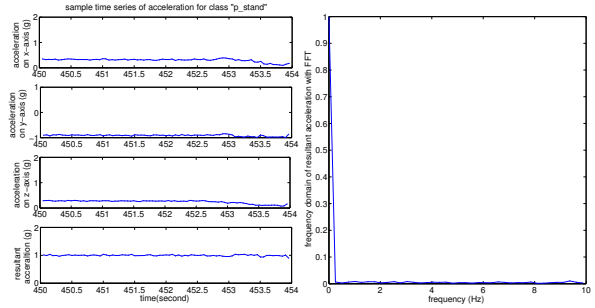
- **Features from RGB-D Camera** The RGB-D cameras get not only RGB images but also the depth information of pixels. Usually, the OpenNI [93] was employed to extract both 2D and 3D bounding boxes of a human subject from raw RGB-D image frames [122]. As shown in Fig 3.5, the coordinates of the centers reflect the detailed location of a human subject, and can be further used to compute displacements and speeds. The coordinates of the corners of bounding boxes can be used to compute the shape of a subject. It is quite straightforward to associate these features with human activities. For example, when the coordinate of the bounding box center moves fast, subjects are more likely to perform motions rather than staying in stationary postures; the shape of a subject is apparently different when the subject



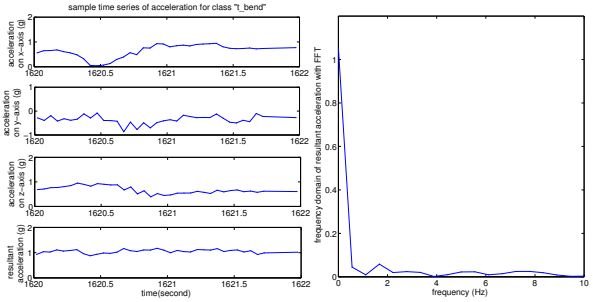
(a) Sample Illustration of Activity Class “a_ascend”



(b) Sample Illustration of Activity Class “a_walk”



(c) Sample Illustration of Activity Class “p_stand”



(d) Sample Illustration of Activity Class “p_bend”

Figure 3.4: Time Domain and Frequency Domain of Activities in acceleration data from [122]. For each subplot: time domain on the left; frequency domain on the right.

Table 3.2: 2D/3D camera features summary.

feature category	2D/3D	feature varaible	features
movement	2D	center coordinate	mean, std, gradient
	3D	center coordinate	mean, std, gradient
shape	2D	length	mean, std
		width	mean, std
		area	mean, std
	3D	length	mean, std
		width	mean, std
		height	mean, std
		volume	mean, std

is standing compared against when he or she is sitting, etc. Based on the above observations, both 2D and 3D movement and shape features have been extracted. The detailed camera features are provided in Table 3.2.

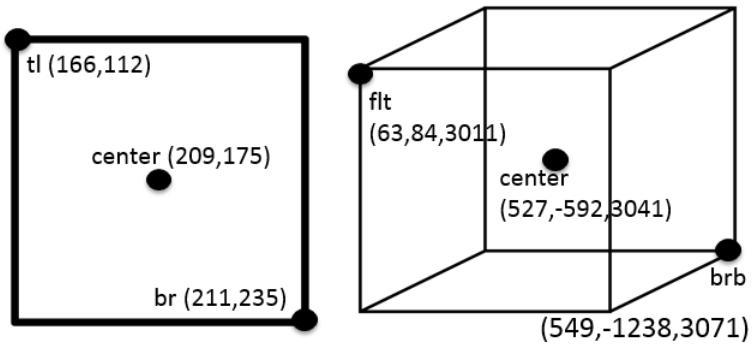


Figure 3.5: The given coordinates of an example bounding box of the RGB-D camera data from [122]. 2D bounding box on the left, 3D bounding box on the right. (tl: top left; br: bottom right; flt: front left top; brb: back right bottom.)

- **Location Feature Extraction** The data gathered by RSSI and PIR sensors is ued for location information. Considering the distinguished layout and decoration of each room in a smart house, the patterns of a same activity in different rooms varies. The detection of the specific room that the subject is located reveals the distinguished patterns in a certain room and brings benefits to improve the classification performance. For instance, a subject is not able to *"p_lie* on the stairs; what's more, the detailed coordinates given by RSSI introduce prior knowledge of a possible activity. For example, the subject is more likely to *"p_lie"* rather than *"a_jump*. The average signal values of RSSI and PIR are computed as location information, while additionally, the standard deviation of RSSI within each second is also computed as an indicator of motion speed. These information will be used in predicting room occupancy for my hierarchical approach.

3.1.3 Annotation Confidence Level

Human activities usually consist of a series of continuous actions, where the transition between one activity to the next is conducted in a graduate manner. Fig 3.2 illustrates an example of a 30-second human activity segment from the labeled *Sphere* data, where the degree of grey scale indicates the confidence score of labeling, or multi-class probability. For instance, in Figure 3.2, *"a_walk"* starts from 2s to 8s. Among all these 7 seconds, only the 5th second has *"a_walk"* with

confidence 1, and during all the rest five seconds, activity *"a_walk"* and *"p_stand"* coexist with different confidence scores (as shown in Table 3.3). This means that the subject conducted a series of standing and walking actions with a gradual and smooth transition between activities. Similar observation for activity of *"t_turn"* exists from 12s to 15s.

Table 3.3: Illustration of activities distribution.

second	2	3	4	5	6	7	8	9
a_walk	0	0.5	0.7	1	0.8	0.7	0.3	0
p_stand	1	0.5	0.3	0	0.2	0.3	0.7	1

3.1.4 Temporal Transitional Dependency

Since there exists significant temporal dependency on the transition among daily activities, it is usually helpful to learn this inherent pattern for activity recognition, especially for transition activities. For example, *t_lie_sit* usually happens in a period between *p_lie* and *p_sit*; *t_bend* usually follows *p_bent*. Some counter examples include that *a_jump* never happens with *p_lie*, and *a_ascend* never happens with *p_squat*, etc.

3.2 Methodology

3.2.1 Multi-Class Learning with Softmax Regression

Softmax regression can be used to compute the posterior probability that the i-th instance in room r belongs to class k as follows [16]:

$$P(Y_i^r = k|\mathbf{X}_{i:}^r) = \frac{\exp(\mathbf{X}_{i:}^r \mathbf{W}_{k:}^r)}{\sum_{s=1}^{K} \exp(\mathbf{X}_{i:}^r \mathbf{W}_{s:}^r)} \equiv \mathbf{P}_{ik}^r, \tag{3.1}$$

where $\mathbf{W}^r \in \Re^{K \times d_r}$ is the model parameter matrix for room r. The parameters can be estimated by minimizing the following cross entropy loss function:

$$\mathbf{W}^r = arg \min_{\mathbf{W}^r} \sum_{i}^{N_r} \sum_{k}^{K} -\mathbf{Y}_{ik}^r \log \mathbf{P}_{ik}^r \tag{3.2}$$

Intuitively, the model produces probabilistic classification results by equation (3.1), and the loss function (3.2) measures the discrepancy between the estimated posterior probability and annotated confidence score of each class for the training examples. The loss function is well-suited for handling soft labels in the human activity recognition problem shown in Fig.3.2, in which multiple activities may occur in the same time period.

3.2.2 Proposed Method: STARS

Although the softmax regression approach can be applied to the smart home data, it has several limitations. First, it does not account for the temporal dependencies among activities in the sequence data. Second, it is designed for learning models independently for each room. Since the amount of training examples available in each room may vary, this may lead to suboptimal local models. Furthermore, the features available to classify the human activities can be different from one room to another. It would be useful to develop a multi-task learning approach that can jointly train the models for all the rooms, taking into account the relationships among the prediction tasks and variable features of the rooms. To overcome these limitations, I propose the following soft multi-task learning framework called STARS, which is designed to optimize the following objective function:

$$
\begin{aligned}
\min_{\Theta} \quad & \mathscr{L}_1 + \mathscr{L}_2 + \mathscr{L}_3 \qquad (3.3)\\
\text{s.t.} \quad & \mathscr{L}_1 = \sum_{r}^{R} \sum_{i}^{N_r} \sum_{k}^{K} -\mathbf{Y}_{ik}^{r} \log \mathbf{P}_{ik}^{r} \\
& \mathscr{L}_2 = \sum_{r}^{R} \beta ||\mathbf{P}^r - \mathbf{G}^r \mathbf{P}^r||_F^2 \\
& \mathscr{L}_3 = \sum_{k}^{K} (\lambda_U ||\mathcal{U}_{k,:,:}||_F + \lambda_V ||\mathcal{V}_{k,:,:}||_F) \\
& \qquad + \sum_{r}^{R} (\lambda_W ||\mathbf{W}^{dif,r}||_F + \lambda_{F1} ||\mathcal{F}^1_{r::}||_F + \lambda_{F2} ||\mathcal{F}^2_{r::}||_F)
\end{aligned}
$$

where,

$$\mathbf{W}^r_{k:} = [\mathcal{W}^{com}_{rk:}, \mathbf{W}^{dif,r}_{k:}]_{1\times d_r},\ \ \mathcal{W}^{com}_{rk:} = \mathcal{U}_{kr:}\mathcal{V}_{k::} \tag{3.4}$$

$$\mathbf{P}^r_{ik} = \frac{\exp(\mathbf{X}^r_{i:}\mathbf{W}^{r\ T}_{k:} + \mathbf{Z}^r_{i-1:}\mathcal{F}^{1\ T}_{1rk:} + \mathbf{Z}^r_{i+1:}\mathcal{F}^{2\ T}_{2rk:})}{\sum_s^K \exp(\mathbf{X}^r_{i:}\mathbf{W}^{r\ T}_{s:} + \mathbf{Z}_{i-1:}\mathcal{F}^{1\ T}_{1rs:} + \mathbf{Z}^r_{i+1:}\mathcal{F}^{2\ T}_{2rs:})} \tag{3.5}$$

$$\mathbf{Z}^r_{i:} = \mathbf{X}^r_{i:}\mathbf{W}^{rT} \tag{3.6}$$

$$\mathbf{G}^r = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \dots & & & & & \dots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}_{N_r\times N_r}$$

where $\Theta = \{\mathcal{U}, \mathcal{V}, \mathbf{W}^r, \mathcal{F}^1, \mathcal{F}^2\}$ corresponds to the set of model parameters for all the rooms $r = 1, 2, \cdots, R$. The framework assumes a linear model for each room r, parameterized by the matrix $\mathbf{W}^r = [\mathbf{W}^{com,r}, \mathbf{W}^{dif,r}]$, where $\mathbf{W}^{com,r}$ is represented by the r-th slice of tensor $\mathcal{W}^{com}$, and denotes the weight matrix associated with the common features for all the rooms. And $\mathbf{W}^{dif,r}$ denotes the weight matrix associated with the unique features of the room. For example, the common features may include those derived from accelerometer sensors worn by the users whereas the unique features may correspond to those derived from surveillance cameras located only in certain rooms.

Note that the objective function consists of three parts: (1) $\mathscr{L}_1$, which is the cross entropy loss function associated with the classification error, (2) $\mathscr{L}_2$, which captures the temporal persistence of the classes (to be explained below), and (3) model complexity control $\mathscr{L}_3$. The posterior probability $\mathbf{P}^r_{ik}$ in the proposed formulation depends not only on the features $\mathbf{X}^r_{i:}$ at time i, but also on the temporal features $\mathbf{Z}_{i-1:}$ and $\mathbf{Z}_{i+1:}$ at time $i-1$ and $i+1$, respectively. We consider $\mathbf{Z}^r_{i-1:} = \mathbf{X}^r_{i-1:}\mathbf{W}^{rT}$ and $\mathbf{Z}^r_{i+1} = \mathbf{X}^r_{i+1:}\mathbf{W}^{rT}$ as temporal features because they are related to the predicted probabilities in the previous and next timesteps. This model also encapsulates information about the class transitions by using the transition tensors $\mathcal{F}^1$ and $\mathcal{F}^2$. Specifically, $\mathcal{F}^1_{r::}$ encodes the relationship between the activity at previous timestep $i-1$ to the activity at current timestep

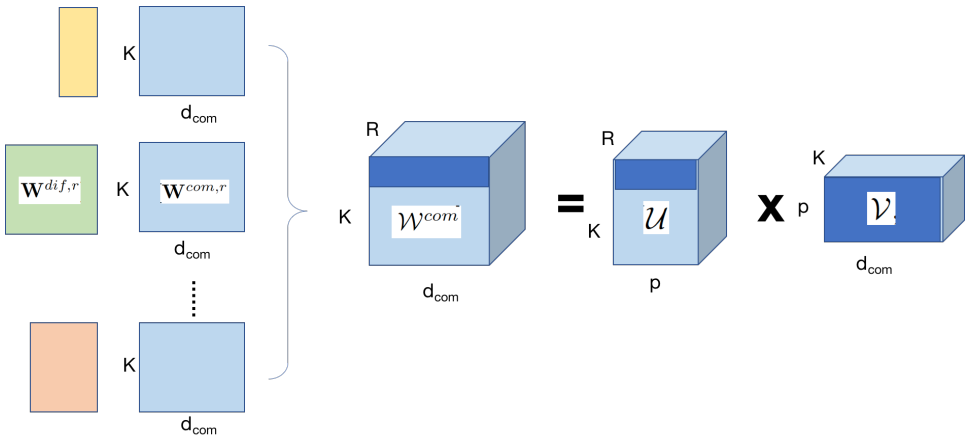


Figure 3.6: The illustration of multi-task learning in STARS

i in room r. Conversely, $\mathcal{F}^2_{r::}$ encodes the relationship between the activity at the next timestep $i+1$ and the activity at current timestep i in room r. These transition tensors are estimated while optimizing the loss function of STARS framework. The second term of the objective function, $\mathscr{L}_2$, is a regularization term to ensure the temporal persistence of the classes. As illustrated in Fig.3.2, most activities tend to last for more than several seconds. This suggests a trivial approach to predict user activity in the next timestep is by using the predicted activity for the current timestep. The temporal persistence of an activity between two adjecent time steps is reflected by the soft constraint $||\mathbf{P}^r_{i:} - \mathbf{P}^r_{i-1:}||^2_F$, where $\mathbf{P}^r_{i-1:} = (\mathbf{G}^r\mathbf{P}^r)_{i:}$. Finally, the third term in the objective function, $\mathscr{L}_3$, is used to control the model complexity to avoid overfitting.

One unique feature of the proposed STARS framework is that it uses a multi-task learning approach to train the models for all rooms simultaneously. Furthermore, instead of treating classification task for different rooms as independent learning problems, it assumes the tasks are related via the common features shared by all the rooms. Fig. 3.6 illustrates how the multi-task works with matrix decomposition. Specifically, although the weight matrix $\mathbf{W}^{com,r}$ for all the rooms can be different, they share a pair of common low-rank factors, $\mathcal{U}$ and $\mathcal{V}$. In Fig. 3.6, we use the notation $\mathcal{W}^{com}$ to represent a 3-dimensional tensor, where the r-th slice of the tensor corresponds to the weight matrix $\mathbf{W}^{com,r}$ for room r.

3.2.3 Optimization

The accelerated gradient descent method can be applied to learn the model parameters. A pseudo-code of the algorithm for both training phase and prediction phase is shown Algorithm 1. The training phase is for inferring the model parameters, and the testing phase is for predicting the next activity in the sequence.

Algorithm 1: STARS Framework

TRAINING PHASE
Input: training set $\{\mathbf{X}^r, \mathbf{Y}^r\}$ and set of regularizers $\{\beta, \lambda_U, \lambda_V, \lambda_W, \lambda_{F1}, \lambda_{F2}\}$.
Output: $\Theta^{(t)} = \{\mathcal{U}^{(t)}, \mathcal{V}^{(t)}, \mathbf{W}^{diff,r(t)}, \mathcal{F}^{1(t)}, \mathcal{F}^{2(t)}\}$
Set $t = 0$ and initialize $\Theta^{(0)} = \{\mathcal{U}^{(0)}, \mathcal{V}^{(0)}, \mathbf{W}^{diff,r(0)}, \mathcal{F}^{1(0)}, \mathcal{F}^{2(0)}\}$.
repeat

$$t = t + 1$$

$$\forall r, q : \mathcal{U}_{qr:} \leftarrow \mathcal{U}_{qr:} - \alpha^{(t)}\left(\frac{\partial \mathscr{L}_1}{\partial \mathcal{U}_{qr:}} + \frac{\partial \mathscr{L}_2}{\partial \mathcal{U}_{qr:}} + \lambda_U \mathcal{U}_{qr:}\right)$$

$$\forall q : \mathcal{V}_{q::} \leftarrow \mathcal{V}_{q::} - \alpha^{(t)}\left(\frac{\partial \mathscr{L}_1}{\partial \mathcal{V}_{q::}} + \frac{\partial \mathscr{L}_2}{\partial \mathcal{V}_{q::}} + \lambda_V \mathcal{V}_{q::}\right)$$

$$\forall r, q : \mathbf{W}^{dif,r}_{q:} \leftarrow \mathbf{W}^{dif,r}_{q:} - \alpha^{(t)}\left(\frac{\partial \mathscr{L}_1}{\partial \mathbf{W}^{dif,r}_{q:}} + \frac{\partial \mathscr{L}_2}{\partial \mathbf{W}^{dif,r}_{q:}} + \lambda_W \mathbf{W}^{dif,r}_{q:}\right)$$

$$\forall r, q : \mathcal{F}^1_{rq:} \leftarrow \mathcal{F}^1_{rq:} - \alpha^{(t)}\left(\frac{\partial \mathscr{L}_1}{\partial \mathcal{F}^1_{rq:}} + \frac{\partial \mathscr{L}_2}{\partial \mathcal{F}^1_{rq:}} + \lambda_{F1} \mathcal{F}^1_{rq:}\right)$$

$$\forall r, q : \mathcal{F}^2_{rq:} \leftarrow \mathcal{F}^2_{rq:} - \alpha^{(t)}\left(\frac{\partial \mathscr{L}_1}{\partial \mathcal{F}^2_{rq:}} + \frac{\partial \mathscr{L}_2}{\partial \mathcal{F}^2_{rq:}} + \lambda_{F2} \mathcal{F}^2_{rq:}\right)$$

until convergence

PREDICTION PHASE
Input: test example, $\mathbf{X}^r_{i:}$, its adjacent predictors, $\mathbf{X}^r_{i-1:}$ and $\mathbf{X}^r_{i+1:}$, and estimated model parameters, $\Theta = \{\mathcal{U}, \mathcal{V}, \mathbf{W}^{dif,r}, \mathcal{F}^1, \mathcal{F}^2\}$, $r = 1, 2, ..., R$
Output: predicted probability
$P(y = k|\mathbf{X}^r_{i:}, \mathbf{X}^r_{i-1:}, \mathbf{X}^r_{i+1:}, \Theta) = \mathbf{P}^r_{ik}$, k = 1,2,...K, with Formula (3.5) and (3.6)

Since there are multiple model parameters, $\Theta = \{\mathcal{U}, \mathcal{V}, \mathbf{W}^{dif,r}, \mathcal{F}^1, \mathcal{F}^2\}$, the parameters are each updated in an alternating fashion. A backtracking line search strategy is also implemented to adaptively choose the step size of the gradient descent [17] and ensure faster convergence. In the remainder of this section, I show the gradient computation of $\mathcal{L}_1$ and $\mathcal{L}_2$ with respect to each model parameter.

- **Gradient Computation for $\mathcal{U}$.**

Taking the partial derivative of $\mathscr{L}_1$ w.r.t. $\mathcal{U}_{qr:}$, where $k = 1, 2, ..., K$, and $q = 1, 2, ..., K$, yields the following:

$$\begin{aligned}\frac{\partial \mathscr{L}_1}{\partial \mathcal{U}_{qr:}} &= \sum_{i}^{N_r}(\mathbf{P}^r_{iq} - \mathbf{Y}^r_{iq})\mathcal{X}^{com}_{ri:}\mathcal{V}^T_{q::} \\ &\quad + \sum_{i}^{N_r}\sum_{k}^{K}(\mathbf{P}^r_{ik} - \mathbf{Y}^r_{ik})(\mathcal{F}^1_{rkq}\mathcal{X}^{com}_{r(i-1):}\mathcal{V}^T_{q::} + \mathcal{F}^2_{rkq}\mathcal{X}^{com}_{r(i+1):}\mathcal{V}^T_{q::}) \end{aligned} \tag{3.7}$$

where $\mathcal{X}^{com}_{ri:}\mathcal{V}^T_{q::}$ denote the latent representation of the common features $\mathcal{X}^{com}_{ri:}$ for room r. The preceding equation suggests that the update formula for $\mathcal{U}_{qr:}$ depends on two terms. The first term on the right hand side of Equation (3.7), $\sum_i^{N_r}(\mathbf{P}^r_{iq} - \mathbf{Y}^r_{iq})\mathcal{X}^{com}_{ri:}\mathcal{V}^T_{q::}$, measures the difference between the predicted and true class in terms of the latent, common feature vectors. The second term, $\sum_i^{N_r}\sum_k^K(\mathbf{P}^r_{ik} - \mathbf{Y}^r_{ik})(\mathcal{F}^1_{rkq}\mathcal{X}^{com}_{r(i-1):}\mathcal{V}^T_{q::} + \mathcal{F}^2_{rkq}\mathcal{X}^{com}_{r(i+1):}\mathcal{V}^T_{q::})$ measures the difference in terms of the latent feature vectors for adjacent time periods, taking into account the temporal dependencies between activities, $\mathcal{F}^1_{r::}$ and $\mathcal{F}^2_{r::}$.

Furthermore, the gradient of $\mathscr{L}_2$ w.r.t. $\mathcal{U}_{qr:}$ is given by

$$\frac{\partial \mathscr{L}_2}{\partial \mathcal{U}_{qr:}} = \sum_{i}^{N}\sum_{k}^{K} 2\beta(\mathbf{P}^r_{ik} - (\mathbf{G}^r\mathbf{P}^r)_{ik}) \times (\frac{\partial \mathbf{P}^r_{ik}}{\partial \mathcal{U}_{qr:}} - \sum_{j}^{N_r}\mathbf{G}^r_{ij}\frac{\partial \mathbf{P}^r_{jk}}{\partial \mathcal{U}_{qr:}})$$

where,

$$\begin{aligned}\frac{\partial \mathbf{P}^r_{ik}}{\partial \mathcal{U}_{qr:}} &= \mathbf{P}^r_{ik}\Bigg((1\{q = k\} - \mathbf{P}^r_{iq})\mathcal{X}^{com}_{ri:}\mathcal{V}^T_{q::} + (\mathcal{F}^1_{rkq}\mathcal{X}^{com}_{r(i-1):}\mathcal{V}^T_{q::} + \mathcal{F}^2_{rkq}\mathcal{X}^{com}_{r(i+1):}\mathcal{V}^T_{q::}) \\ &\quad - \sum_{s}^{K}(\mathcal{F}^1_{rsq}\mathcal{X}^{com}_{r(i-1):}\mathcal{V}^T_{q::} + \mathcal{F}^2_{rsq}\mathcal{X}^{com}_{r(i+1):}\mathcal{V}^T_{q::})\mathbf{P}^r_{is}\Bigg)\end{aligned}$$

- **Gradients Computation for $\mathcal{V}$.** Similarly, the gradients w.r.t. $\mathcal{V}$ are:

$$\begin{aligned}\frac{\partial \mathscr{L}_1}{\partial \mathcal{V}_{q::}} &= \sum_{r}^{R}\sum_{i}^{N_r}(\mathbf{P}^r_{iq} - \mathbf{Y}^r_{iq})\mathcal{U}_{qr:}\mathcal{X}^{com}_{ri:} \\ &\quad + \sum_{r}^{R}\sum_{i}^{N_r}\sum_{k}^{K}(\mathbf{P}^r_{ik} - \mathbf{Y}^r_{ik})(\mathcal{F}^1_{rkq}\mathcal{U}_{qr:}\mathcal{X}^{com}_{r(i-1):} + \mathcal{F}^2_{rkq}\mathcal{U}_{qr:}\mathcal{X}^{com}_{r(i+1):}) \\ \frac{\partial \mathscr{L}_2}{\partial \mathcal{V}_{q::}} &= \sum_{r}^{R}\sum_{i}^{N_r}\sum_{k}^{K} 2\beta(\mathbf{P}^r_{ik} - (\mathbf{G}^r\mathbf{P}^r)_{ik}) \times (\frac{\partial \mathbf{P}^r_{ik}}{\partial \mathcal{V}_{q::}} - \sum_{j}^{N_r}\mathbf{G}^r_{ij}\frac{\partial \mathbf{P}^r_{jk}}{\partial \mathcal{V}_{q::}})\end{aligned}$$

- **Gradients Computation for $\mathbf{W}^{dif,r}$.**

$$\frac{\partial \mathscr{L}_1}{\partial \mathbf{W}^{dif,r}_{q:}} = \sum_{i}^{N_r}(\mathbf{P}^r_{iq} - \mathbf{Y}^r_{iq})\mathbf{X}^{dif,r}_{i:} + \sum_{i}^{N_r}\sum_{k}^{K}(\mathbf{P}^r_{ik} - \mathbf{Y}^r_{ik})(\mathcal{F}^1_{rkq}\mathbf{X}^{dif,r}_{i-1:} + \mathcal{F}^2_{rkq}\mathbf{X}^{dif,r}_{i+1:})$$

$$\frac{\partial \mathscr{L}_2}{\partial \mathbf{W}^{dif,r}_{q:}} = \sum_{i}^{N_r}\sum_{k}^{K} 2\beta(\mathbf{P}^r_{ik} - (\mathbf{G}^r\mathbf{P}^r)_{ik}) \times (\frac{\partial \mathbf{P}^r_{ik}}{\partial \mathbf{W}^{dif,r}_{q:}} - \sum_{j}^{N_r}\mathbf{G}^r_{ij}\frac{\partial \mathbf{P}^r_{jk}}{\partial \mathbf{W}^{dif,r}_{q:}})$$

- **Gradients Computation for $\mathcal{F}^1$ (or $\mathcal{F}^2$).**

$$\frac{\partial \mathscr{L}_1}{\partial \mathcal{F}^1_{rq:}} = \sum_{i}^{N_r}(\mathbf{P}^r_{iq} - \mathbf{Y}^r_{iq})\mathbf{Z}^r_{i-1:}$$

$$\frac{\partial \mathscr{L}_2}{\partial \mathcal{F}^1_{rq:}} = 2\beta\sum_{i}^{N_r}\sum_{k}^{K}(\mathbf{P}^r_{ik} - (\mathbf{G}^r\mathbf{P}^r)_{ik}) \times (\frac{\partial \mathbf{P}^r_{ik}}{\partial \mathcal{F}^1_{rq:}} - \sum_{j}^{N_r}\mathbf{G}^r_{ij}\frac{\partial \mathbf{P}^r_{jk}}{\partial \mathcal{F}^1_{rq:}})$$

The gradients $\frac{\partial \mathscr{L}_1}{\partial \mathcal{F}^2_{rq:}}$ and $\frac{\partial \mathscr{L}_2}{\partial \mathcal{F}^2_{rq:}}$ can be obtained in a similar way.

3.3 Experimental Evaluation

I performed the experiments on a real-world data set from the *SPHERE Challenge* competition [122]. The dataset contains classes of human activities recorded in a house with 9 rooms. The raw data contains 10 sequences, where each sequence corresponds to a series of activities performed by a subject for a time period lasting between $1,392$ to $1,825$ seconds. With the sensor observations at each second as a data instance, I ended up with a total of $16,124$ instances. Each instance was labeled by a team of 12 annotators [122], whose results are aggregated to obtain a confidence score for each class label. For evaluation purposes, I apply 5-fold cross-validation and report the mean and standard deviation of their prediction accuracies.

Following the approach described in [122], I employ the weighted Brier score to evaluate the classification performance. The metric is defined as follows:

$$\text{BS} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} l_k(\mathbf{Y}_{ik} - \mathbf{P}_{ik})^2 \tag{3.8}$$

where $\mathbf{Y}_{ik}$ is the confidence score for the i-th instance and k-th class, computed based on the labels provided by a team of annotators, whereas $\mathbf{P}_{ik}$ is the predicted posterior class. The weight for each class, l_k, is defined in [3], which is negatively correlated with the class size.

3.3.1 Baseline Algorithms

I compare the performance of my proposed framework, **STARS**, against the following baseline algorithms:

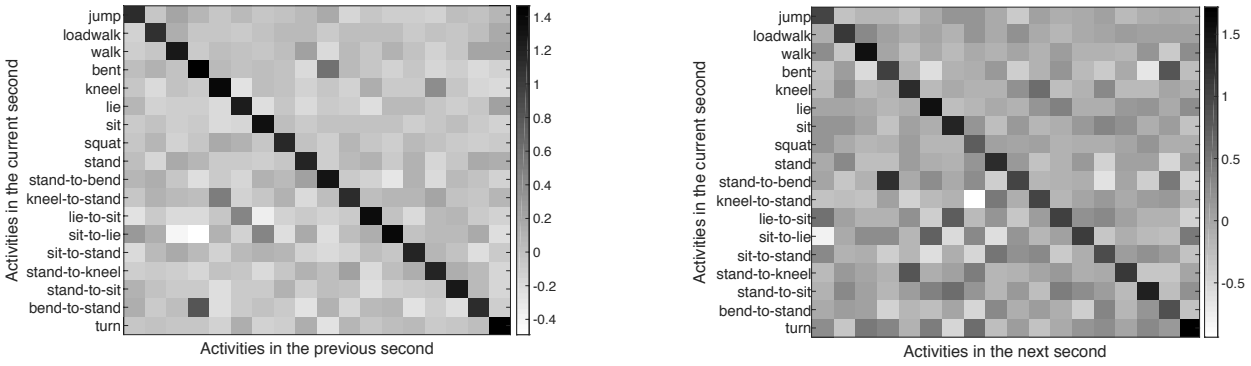
- **SR**: Softmax regression, which trains a local softmax regression model for each room with Equation (3.1) and (3.2). Unlike my proposed framework, it is a single-task learning model and does not incorporate temporal dependencies.
- **KNN**: A k-nearest neighbor classifier, which is another baseline used in [3] for the SPHERE competition data. I sum up the weights associated with each neighboring instance $\mathbf{Y}_{ik}$ for the test instance i and normalize the weighted sum to obtain the predicted posterior probabilities.
- **CRF**: Conditional Random Field [118], which is a widely used model for sequence classification problems [45, 65] and has been applied to activity recognition problems [123, 59].

3.3.2 Experimental Results

The results comparing the weighted brier score of the proposed framework STARS, against the baseline methods (SR, KNN and CRF) are shown in Table 3.4. I reported the weighted Brier score for all rooms (denoted as Overall) as well as for individual rooms. The results suggest that the overall performance of STARS is significantly better than the baseline methods. In terms of performances for individual rooms, STARS achieves the best (i.e., lowest score) in 7 out of the 9 rooms. The performance of STARS is slightly worse than SR for `bedroom2` and `hallway` due to the lack of transitional activities, making it harder to learn the temporal dependency accurately based on their limited training data.

Table 3.4: Weighted brier scores for various competing algorithms.

Room	SR	KNN	CRF	STARS
bathroom	0.1334±0.0149	0.1315±0.0103	0.1479±0.0152	**0.1269**±0.0173
bedroom1	**0.0853**±0.0129	0.1026±0.0085	0.0935±0.0118	0.0920±0.0156
bedroom2	0.2817±0.0148	0.2862±0.0178	0.2886±0.0223	**0.2675**±0.0172
hallway	**0.1926**±0.0953	0.2323±0.0675	0.2115±0.0816	0.1953±0.0849
kitchen	0.0842±0.0122	0.0915±0.0099	0.0917±0.0133	**0.0820**±0.0106
living room	0.1594±0.0181	0.1774±0.0171	0.1710±0.0201	**0.1468**±0.0142
stairs	0.3827±0.0705	0.4366±0.0552	0.3834±0.0288	**0.3373**±0.0505
study room	0.0441±0.0407	0.0649±0.0240	0.0506±0.0556	**0.0381**±0.0365
toilet	0.1440±0.0380	0.1368±0.0304	0.1442±0.0358	**0.1360**±0.0417
Overall	0.1700±0.0095	0.1815±0.0089	0.1794±0.0121	**0.1598**± 0.0087



(a) Transition matrix from $(i-1)$ to i timestep.

(b) Transition matrix from i to $(i+1)$ timestep.

Figure 3.7: The estimated transition matrices $\mathcal{F}^1_{r::}$ (left) and $\mathcal{F}^2_{r::}$ (right) for `living room`. The ordering of the classes on the horizontal and vertical axes are the same.

In addition to its lower brier score, another advantage of STARS framework is that the model can be used to learn the transition between activities via $\mathcal{F}^1$ and $\mathcal{F}^2$ as by-product. Fig. 3.7a and 3.7b depict a heat map of the two tensor slices $\mathcal{F}^1_{r::}$ and $\mathcal{F}^2_{r::}$ for the `living room`. The results shown in these figures are mostly consistent with our common sense knowledge. For example, Fig.3.7a shows that the `bent` posture is mostly followed by the activity `bend-to-stand` whereas `stand-to-bend` often leads to the `bent` posture. Similarly, Fig.3.7b shows that the `stand-to-kneel` activity would lead to the `kneel` posture in the next time step, while `lie-to-sit` begins with `lie` posture and ends with the `sit` posture.

3.4 Related Work

Numerous approaches have been developed for human activity recognition. Classic approaches include decision tree [19], support vector machine [128], logistic regression, and Bayesian networks [36]. For example, a comparison between logistic regression and non-linear SVM on human activity recognition is given in [72]. These approaches do not consider the temporal/sequential dependencies between activities. In contrast, methods such as Hidden Markov Model (HMM) and Conditional Random Fields (CRF) are more well-suited for handling sequential data [45, 142], and thus, have been widely utilized for activity recognition tasks [67, 123]. However, these approaches are primarily designed for single-task learning, unlike the multi-task approach proposed in this study. The success of multi-task learning for activity recognition has been well-documented [117, 140, 141, 7, 22]. In [117], a structured multi-task classification method was proposed, where each task corresponds to the classification of a specific person. [140] presented a multi-task clustering framework for analyzing daily living activities from visual data collected by wearable cameras. In addition, [141] focused on multi-task feature selection whereas [7] focused on online matrix regularization. Unlike other existing works, my proposed framework considers the classification in different rooms as separate tasks, with possibly different types of features.

3.5 Summary

In this thesis, I present a soft multi-task learning technique for human activity recognition from multi-modal sensor data in a smart home. The proposed technique incorporates the temporal dependencies between classes in a multi-task learning setting. Experimental results using a public human activity recognition dataset showed that the proposed technique outperforms baseline methods including K-Nearest Neighbor, Conditional Random Field, and single-task learning with multinomial softmax regression. The framework not only improves the classification performance, it also reveals the typical type of transitions between activities.

CHAPTER 4

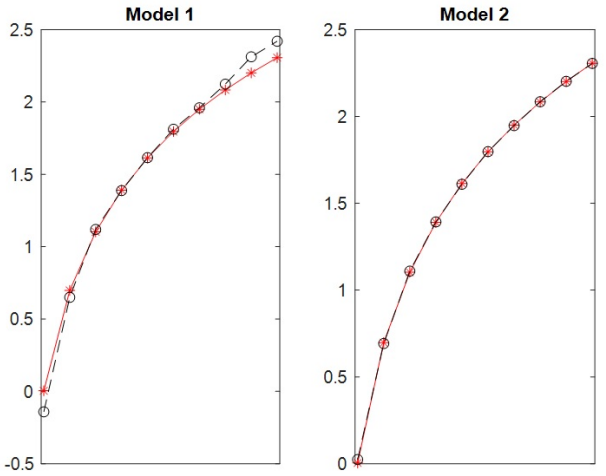
DISTRIBUTION PRESERVING MULTI-TASK REGRESSION FOR SPATIO-TEMPORAL DATA

Regression methods play an important role in many spatio-temporal applications as they can be used to solve a wide variety of prediction problems such as projecting future changes in the climate system, predicting the crime rate in urban cities, or forecasting traffic volume on highways. Although accuracy is an important requirement, building models that can replicate the future distribution of data is just as important since the predicted distribution can be used for planning, risk assessment, and other decision making purposes. For example, in climate modeling, knowing the changes in future distribution of climate variables such as temperature and precipitation can help scientists to better estimate the severity and frequency of adverse weather events in the future. In agricultural production, the predicted distribution can be used to derive statistics such as average length of future growing season or persistence of wet and dry spells, which are important metrics for farmers and agricultural researchers.

However, previous studies have shown that the distribution of predicted values generated by traditional regression methods are not always consistent with the true distribution of the data even when the prediction errors are relatively low [5, 6]. While distribution-preserving methods such as quantile mapping [120] have been developed to overcome this limitation, their prediction errors can still be high [5]. For example, Figure 4.1(a) shows a comparison between the predicted values of a non-distribution preserving model (Model 1) against a distribution-preserving model (Model 2) on a set of 10 values. Although the first model has a much lower root mean square error (RMSE) it does not fit well the tails of the predicted distribution, as shown in Figure 4.1, compared to the second model, which fits the distribution almost perfectly but has a considerably higher RMSE. This has led to the growing interest in developing techniques that can minimize both prediction error and the divergence between the true and predicted distributions [5, 6]. However, current techniques are mostly designed for single task learning problems, i.e., to build a regression model for a single

True Value	Model 1	Model 2
0.0043	-0.1411	0.6924
0.6974	0.6501	0.0241
1.1029	1.1194	1.1095
1.3906	1.3881	1.6097
1.6137	1.6157	1.3931
1.7960	1.8120	1.7977
1.9502	1.9603	1.9473
2.0837	2.1237	2.2026
2.2015	2.3125	2.0850
2.3069	2.4202	2.3059
RMSE	0.0713	0.3243

(a) Predicted values



(b) Cumulative distribution function of predicted values

Figure 4.1: Comparison between the predictions of non-distribution preserving (Model 1) and distribution preserving (Model 2) methods in terms of their root mean squared errors and cumulative distribution functions.

location. For multi-location prediction, these models are trained independently, and thus, often fail to capture the inherent autocorrelations of the spatio-temporal data. In addition, their accuracy and distribution fit are likely to be suboptimal for locations with limited training data. Therefore, multi-task learning algorithms should be developed for multi-location problems (e.g. [134, 139, 136]).

To account for spatial autocorrelation and the imbalanced distribution of training data, there have been several recent studies focusing on the development of multi-task learning (MTL) methods [148] [147] for spatio-temporal data [137] [136]. MTL learns a local model for each location, but leverages data from other locations to improve its model performance. It accomplishes this by assuming that the local models share some common structure, which can be exploited to enhance their predictive performance. Unfortunately, existing MTL approaches are mostly focused on minimizing the residual error, paying scant attention to how realistic is the overall predicted distribution.

This chapter presents a novel distribution-preserving multi-task learning framework for spatio-temporal data. Our framework assumes that the local models share a common low-rank representation, similar to the assumption used in [136]. It also employs a graph Laplacian regularizer based on the Haversine spatial distance to preserve the spatial autocorrelation in the data. A non-parametric kernel density estimation approach with L^2-distance is used to determine the divergence between

the predicted and true distributions of the data. Both the distribution fitting and multi-task learning are integrated into a unified objective function, which is optimized using a mini-batch accelerated gradient descent algorithm. Experimental results using a real-world climate dataset from the Global Historical Climatology Network (GHCN) showed that our proposed framework outperformed the non-distribution preserving approaches in more than 78% of the weather stations considered in this study, with an average reduction in distribution error between 7.5% – 17.8%. Our approach also outperforms a distribution-preserving single-task regression method called contour regression [5] in at least 78% of the weather stations.

4.1 Preliminaries

Our framework is designed not only to learn accurate local regression models, but also to generate predictions that are consistent with the true distribution of the data. This requires an approach for estimating the density function of the response variable and a divergence measure to compute the difference between two distributions. We review these approaches in this section.

4.1.1 Density Estimation

There are various density estimation methods that have been proposed in the literature. In general, these methods can be divided into two categories:

- **Parametric Methods**, which assume that the density function follows certain parametric distribution, such as Gaussian, Gamma, exponential and so on. The sampled data points are used to estimate parameters of the distribution. As the number of parameters tends to be small, parametric methods has an advantage in that they do not require a large number of points to fit the density function. Unfortunately, many real-world datasets may not follow the standard distributions as they often comprise of complex mixtures of distributions, which leads to imprecise estimation of their true distribution.

- **Non-parametric Methods**, which avoid making *a priori* assumption about the shape of the

distributions. Two popular methods are the K-nearest neighbor (KNN) approach and kernel density estimation. The KNN approach uses only the k nearest neighbors to estimate the density function whereas the kernel density estimation (KDE) approach uses a mixture of Gaussian distributions centered at all the data points with a smoothing kernel width parameter, h.

Due to the generality of the approach for fitting unknown distribution, we employ the KDE approach to approximate the density function. Given N data points sampled from an unknown distribution of a variable Y , $\mathbf{y} = \{y_i|, i = 1, 2, ..., N\}$, the density function of Y is estimated as follows:

$$P_{\mathbf{y}}(Y = y) = \frac{1}{N} \sum_{i}^{N} G(y|y_i, h^2) \tag{4.1}$$

4.1.2 Divergence Measures

A divergence measure can be used to estimate the difference between two density functions. This section reviews some of the divergence measures used in this chapter. Although there are other measures available, evaluating them is beyond the scope of this chapter and is a subject for future research.

4.1.2.1 RMS-CDF

RMS-CDF is a measure defined in [5] to compare two empirical cumulative distribution functions. Let $\mathbf{y}$ and $\hat{\mathbf{y}}$ denote vectors of length N sampled from two distributions. The RMS-CDF measure between the pair of distributions is computed as follows:

$$\text{RMS-CDF} = \sqrt{\frac{1}{N} \sum_{i}^{N} (y_{(i)} - \hat{y}_{(i)})^2} \tag{4.2}$$

where $y_{(i)}$ represents the i-th largest value in $\mathbf{y}$ and $\hat{y}_{(i)}$ is the i-th largest value in $\hat{\mathbf{y}}$. The measure is obtained by sorting the values in $\mathbf{y}$ and $\hat{\mathbf{y}}$ and computing the average sum of squared difference between their sorted values.

4.1.2.2 L^2 Distance

Given a pair of random variables, Y and $\hat{Y}$, along with their respective probability density functions, P_Y and $P_{\hat{Y}}$, their L^2-Distance can be calculated as follows [115]:

$$L^2(P_Y, P_{\hat{Y}}) = \int (P_Y(y) - P_{\hat{Y}}(y))^2 dy$$

Unlike RMS-CDF, L^2-Distance measures the divergence of two distribution based on probability density functions instead of their cumulative distribution functions.

4.2 Proposed Framework

This section introduces our proposed framework for distribution-preserving multi-task regression. The framework uses kernel density estimation (KDE) to estimate the probability density function. KDE provides a flexible approach for modeling the density function of unknown distributions unlike other parametric approaches. We also employ an L^2-Distance to measure the difference between two density functions.

4.2.1 Divergence Measurement

In this chapter, we use kernel density estimation (KDE) to estimate the probability density function and the L^2-Distance to measure the divergence between two density functions. Given N data points sampled from an unknown distribution of a variable Y , $\mathbf{y} = \{y_i|, i = 1, 2, ..., N\}$, the density function of Y is estimated as follows:

$$P_{\mathbf{y}}(Y = y) = \frac{1}{N} \sum_{i}^{N} G(y|y_i, h^2) \tag{4.3}$$

where, $G(y|\mu, \sigma)$ represents Gaussian kernel with mean μ and standard deviation σ, and h is the Parzan window width.

We now derive our approach for computing the divergence between two estimated probability distributions, using the Gaussian kernel density estimator with L^2-Distance. Let Y be a random variable. Consider two N-dimensional vectors $\mathbf{y} = [y_1, y_2, ...y_N]^T$ and $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, ...\hat{y}_N]^T$, where

Table 4.1: Summary of notations used in the chapter.

Notation	Definition
S	number of stations(tasks)
N_s	number of observations in task s
d	number of features
k	number of latent factors, $k < d$
$\mathbf{X}_s \in \Re^{N_s \times d}$	predictor matrix for station s
$\mathbf{y}_s$ or $\hat{\mathbf{y}}_s \in \Re^{N_s \times 1}$	observed or estimated response samples at station s
$P_{\mathbf{y}}(Y)$ or $P_{\hat{\mathbf{y}}}(Y)$	density function of Y estimated by observed(or estimated) samples
$\mathbf{W} \in \Re^{d \times S}$	model parameter
$\mathbf{U} \in \Re^{d \times k}$	latent factors
$\mathbf{V} \in \Re^{k \times S}$	linear coefficients for latent factors
h, $\hat{h}$	Parzen window widths for $\mathbf{y}$ and $\hat{\mathbf{y}}$
$G(y\|\mu, \sigma)$	Gaussian distribution
$\mathbf{Q} \in \Re^{S \times S}$	pairwise Haversine distances
$\mathbf{A} \in \Re^{S \times S}$	adjacency matrix, where $A_{ij} = \frac{1}{\exp(Q_{ij}/\gamma)}$
$\mathbf{D} \in \Re^{S \times S}$	a diagonal matrix, $D_{ii} = \sum_j A_{ij}$

the y_i's and $\hat{y}_j$'s are randomly drawn from the sample space of Y. The density functions for Y estimated using Gaussian KDE from the two sample vectors, $P_{\mathbf{y}}(Y)$ and $P_{\hat{\mathbf{y}}}(Y)$, can be written as:

$$P_{\mathbf{y}}(Y = y) = \frac{1}{N} \sum_{i}^{N} G(y|y_i, h^2)$$

$$P_{\hat{\mathbf{y}}}(Y = y) = \frac{1}{N} \sum_{i}^{N} G(y|\hat{y}_i, \hat{h}^2)$$

The following theorem presents the closed-form formula for computing the L^2-Distance between two estimated density functions.

Theorem 1 *L^2-Distance between $P_{\mathbf{y}}(Y)$ and $P_{\hat{\mathbf{y}}}(Y)$ is:*

$$\begin{aligned} L^2(P_{\mathbf{y}}, P_{\hat{\mathbf{y}}}) &= \int (P_{\mathbf{y}}(y) - P_{\hat{\mathbf{y}}}(y))^2 dy \\ &= \frac{1}{N^2} \sum_{i,j}^{N} \left[G(y_i|y_j, 2h^2) + G(\hat{y}_i|\hat{y}_j, 2\hat{h}^2) - 2G(y_i|\hat{y}_j, h^2 + \hat{h}^2) \right] \end{aligned}$$

Proof: We begin by expressing the L^2-Distance in terms of their KDE functions:

$$\begin{aligned} L^2(P_{\mathbf{y}}, P_{\hat{\mathbf{y}}}) &= \int (P_{\mathbf{y}}(y) - P_{\hat{\mathbf{y}}}(y))^2 dy \\ &= \int \left(\frac{1}{N} \sum_{i}^{N} G(y|y_i, h^2) - \frac{1}{N} \sum_{i}^{N} G(y|\hat{y}_i, \hat{h}^2) \right)^2 dy \end{aligned}$$

Expanding the square yields the following expression:

$$\begin{aligned} L^2(P_{\mathbf{y}}, P_{\hat{\mathbf{y}}}) &= \frac{1}{N^2} \sum_{i,j}^{N} \int G(y|y_i, h^2) G(y|y_j, h^2) dy \\ &+ \frac{1}{N^2} \sum_{i,j}^{N} \int G(y|\hat{y}_i, \hat{h}^2) G(y|\hat{y}_j, \hat{h}^2) dy \\ &- \frac{2}{N^2} \sum_{i,j}^{N} \int G(y|y_i, h^2) G(y|\hat{y}_j, \hat{h}^2) dy \end{aligned} \tag{4.4}$$

Based on the fact that the product of two Gaussian distributions, $G(y|\mu_1, \sigma_1^2)$ and $G(y|\mu_2, \sigma_2^2)$, can be written as follows [18]:

$$G(y|\mu_1, \sigma_1^2) G(y|\mu_2, \sigma_2^2) = G(\mu_1|\mu_2, \sigma_1^2 + \sigma_2^2) \times G(y|\mu_{12}, \sigma_{12}),$$

where $\mu_{12} = \frac{\mu_1 \sigma_2^2 + \mu_2 \sigma_1^2}{\sigma_2^2 + \sigma_1^2}, \sigma_{12} = \sqrt{\frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}}$.
Since $\int G(y|\mu, \sigma) = 1$, the integral for the product of two Gaussians can be written as:

$$\int G(y|\mu_1, \sigma_1^2) G(y|\mu_2, \sigma_2^2) dy = G(\mu_1|\mu_2, \sigma_1^2 + \sigma_2^2)$$

Replacing this into Equation (4.4) leads to the following;

$$\frac{1}{N^2} \sum_{i,j}^{N} \left(G(y_i|y_j, 2h^2) + G(\hat{y}_i|\hat{y}_j, 2\hat{h}^2) - 2G(y_i|\hat{y}_j, h^2 + \hat{h}^2) \right),$$

which completes the proof. □

4.2.2 DPMTL: Distribution-Preserving MTL Framework

Let $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ be a spatial-temporal dataset, where $\mathcal{X} = \{\mathbf{X}_s | s = 1, 2, ..., S\}$, $\mathcal{Y} = \{\mathbf{y}_s |, s =$

$1, 2, ...S\}$, and S is the number of locations. Each $\mathbf{X}_s \in \mathbb{R}^{N_s \times d}$ is a d-dimensional multivariate time series of predictor variables at location s, and each $\mathbf{y}_s \in \mathbb{R}^{N_s}$ is the time series for observations at location s. Furthermore, N_s is the number of training examples available at location s. The motivation for our spatial-temporal distribution preserving approach is to learn a set of local linear models, $f_s(\mathbf{x}; \mathbf{w}_s)$ that minimizes the prediction error while fitting the marginal distribution of Y.

Learning the local models amounts to estimating the weight matrix $\mathbf{W} = [\mathbf{w}_1\ \mathbf{w}_2 \cdots \mathbf{w}_S]$ for all the stations. Due to the inherent relationships between the prediction tasks at multiple locations, our proposed framework assumes that $\mathbf{W} \in \mathbb{R}^{d \times S}$ is not a full rank matrix and can be decomposed into a product of two low-rank matrices, $\mathbf{U} \in \mathbb{R}^{d \times k}$ and $\mathbf{V} \in \mathbb{R}^{k \times S}$, where $k < d$. These low-rank matrices can be derived as follows:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V}} \quad & \mathcal{L}_1 + \alpha \mathcal{L}_2 + \lambda \mathcal{L}_3, \\ \text{s.t.} \quad & \mathbf{W} = \mathbf{U}\mathbf{V},\ \hat{\mathbf{y}}_s = \mathbf{X}_s \mathbf{w}_s \end{aligned} \tag{4.5}$$

where

$$\begin{aligned} \mathcal{L}_1 &= \sum_{s}^{S} ||\mathbf{y}_s - \hat{\mathbf{y}}_s)||_2^2 \\ \mathcal{L}_2 &= \mathbf{tr}[\mathbf{W}(\mathbf{D} - \mathbf{A})\mathbf{W}^T)] \\ \mathcal{L}_3 &= \sum_{s}^{S} \left\{ \frac{1}{N_s^2} \sum_{i,j}^{N_s} \left(G(y_{si}|y_{sj}, 2h_s^2) \right.\right. \\ &\qquad \left.\left. + G(\hat{y}_{si}|\hat{y}_{sj}, 2\hat{h}_s^2) - 2G(y_{si}|\hat{y}_{sj}, h_s^2 + \hat{h}_s^2) \right) \right\} \end{aligned} \tag{4.6}$$

Our objective function consists of three loss functions: (1) $\mathcal{L}_1$, which measures the residual errors on the training data between ground truth samples in $\mathbf{y}_s$ and estimated samples in $\hat{\mathbf{y}}_s$, (2) $\mathcal{L}_2$, which is a regularizer to ensure that the model parameters for two neighboring locations should be close to each other, and (3) $\mathcal{L}_3$, which measures the divergence between the true and predicted distributions for Y.

For $\mathcal{L}_2$, an $S \times S$ similarity matrix $\mathbf{A}$ is calculated by applying an RBF kernel on the Haversine distance [126], Q_{ij}, between two locations i and j, i.e., $A_{ij} = \exp[-Q_{ij}/\gamma]$. $\mathbf{W} \in \mathbb{R}^{d \times S}$ is the

model parameter matrix, where each column corresponds to the weights of the regression model for a given station. The second loss function ensures that spatial autocorrelation is preserved by our framework (following Tobler's First Law of Geography [87]). It accomplishes this by using a graph Laplacian regularizer, where $\mathbf{D}$ is a diagonal matrix whose diagonal elements are $D_{ii} = \sum_j A_{ij}$.

The first loss function $\mathcal{L}_1$ is designed to minimize prediction error by learning the conditional probability function $P_{\mathbf{y}_s}(Y|\mathbf{X})$. In contrast, the third loss function, $\mathcal{L}_3$, is designed to accurately fit the marginal distribution $P_{\mathbf{y}_s}(Y)$ by minimizing the L^2-Distance between the true and estimated density functions for all stations using Equation (4.6). The hyperparameter λ is used to control the trade-off between minimizing the two loss functions. Note that the local models $\mathbf{w}_s$ are assumed to share a common latent matrix $\mathbf{U}$. Such an assumption is useful especially for learning models at locations with limited training data. Specifically, the model parameters for each station s are assumed to be formed using a linear combination of the dictionary (latent factors) in $\mathbf{U}$, with $\mathbf{v}_s$(the s-th column of $\mathbf{V}$) specifying the coefficients of the linear combination.

4.2.3 Optimization

We employ a mini-batch accelerated gradient descent approach to solve the optimization problem given in Equation (4.5). This requires us to derive the gradient of each term, $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$, with respect to the model parameters.

For $\mathcal{L}_1$, the partial derivatives are given by:

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{U}} = \sum_s^S 2\mathbf{X}_s^T\mathbf{X}_s\mathbf{U}\mathbf{v}_s\mathbf{v}_s^T - 2\mathbf{X}_s^T\mathbf{y}_s\mathbf{v}_s^T \tag{4.7}$$

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{v}_s} = 2\mathbf{U}^T\mathbf{X}_s^T\mathbf{X}_s\mathbf{U}\mathbf{v}_s - 2\mathbf{U}^T\mathbf{X}_s^T\mathbf{y}_s \tag{4.8}$$

For $\mathcal{L}_2$, the partial derivatives are given by:

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{U}} = 2\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T \tag{4.9}$$

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{V}} = 2\mathbf{U}^T\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A}) \tag{4.10}$$

For $\mathcal{L}_3$, we first show the partial derivative of L^2-Distance with respect to $\hat{y}_{sm}$. The L^2-Distance of each station, given in Equation (4.6), is composed of three Gaussian kernels. Since $G(y_{si}|y_{sj}, 2h^2)$ is a constant with respect of $\hat{y}_{sm}$, its partial derivative is:.

$$\sum_{i,j}^{N_s} \frac{\partial G(y_{si}|y_{sj}, 2h^2)}{\partial \hat{y}_{sm}} = 0$$

The derivative for the sum of pairwise Gaussian kernel of $\hat{y}_s$ is given as follows:

$$\begin{aligned}
&\sum_{i,j}^{N_s} \frac{\partial G(\hat{y}_{si}|\hat{y}_{sj}, 2\hat{h}^2)}{\partial \hat{y}_m} \\
= \;& \frac{-1}{2\hat{h}^2} G(\hat{y}_{si}|\hat{y}_{sj}, 2\hat{h}^2) \times \Big[1\{i = m\} - 1\{j = m\}\Big](\hat{y}_{si} - \hat{y}_{sj}) \\
= \;& \sum_{j}^{N_s} \frac{-1}{2\hat{h}^2} G(\hat{y}_{sm}|\hat{y}_{sj}, 2\hat{h}^2)(\hat{y}_{sm} - \hat{y}_{sj}) - \sum_{i}^{N_s} \frac{-1}{2\hat{h}^2} G(\hat{y}_{si}|\hat{y}_{sm}, 2\hat{h}^2)(\hat{y}_{si} - \hat{y}_{sm}) \\
= \;& \frac{-1}{\hat{h}^2} \sum_{i}^{N_s} G(\hat{y}_{sm}|\hat{y}_{si}, 2\hat{h}^2)(\hat{y}_{sm} - \hat{y}_{si})
\end{aligned}$$

The derivative for the cross-term Gaussian kernels is

$$\begin{aligned}
&\sum_{i,j}^{N_s} \frac{\partial G(\hat{y}_{si}|y_{sj}, h^2 + \hat{h}^2)}{\partial \hat{y}_{sm}} \\
= \;& \sum_{i,j}^{N_s} \frac{-1}{h^2 + \hat{h}^2} G(\hat{y}_{si}|y_{sj}, h^2 + \hat{h}^2) \times 1\{i = m\}(\hat{y}_{si} - y_{sj}) \\
= \;& \frac{-1}{h^2 + \hat{h}^2} \sum_{i}^{N_s} G(\hat{y}_{sm}|y_{si}, h^2 + \hat{h}^2)(\hat{y}_{sm} - y_{si})
\end{aligned}$$

Putting together all three derivatives, the partial derivative of $\mathcal{L}_3$ w.r.t. $\hat{y}_{sm}$ is given by

$$\begin{aligned}
\frac{\partial \mathcal{L}_3}{\partial \hat{y}_{sm}} = \;& \frac{1}{{N_s}^2} \Bigg[\frac{2}{h^2 + \hat{h}^2} \sum_{i}^{N_s} G(\hat{y}_{sm}|y_{si}, h^2 + \hat{h}^2)(\hat{y}_{sm} - y_{si}) \\
& - \frac{1}{\hat{h}^2} \sum_{i}^{N_s} G(\hat{y}_{sm}|\hat{y}_{si}, 2\hat{h}^2)(\hat{y}_{sm} - \hat{y}_{si}) \Bigg]
\end{aligned}$$

Furthermore, denoting $\mathbf{x}_{sm} \in \Re^{d \times 1}$ (the m-th row of $\mathbf{X}_s$), the partial derivative of $\hat{y}_{sm}$ with respect to $\mathbf{U}$ and $\mathbf{V}$ are:

$$\frac{\partial \hat{y}_{sm}}{\partial \mathbf{U}} = \mathbf{x}_{sm}^T \mathbf{v}_s^T \in \mathbb{R}^{d \times k}, \quad \frac{\partial \hat{y}_{sm}}{\partial \mathbf{v}_s} = \mathbf{U}^T \mathbf{x}_{sm}^T \in \mathbb{R}^{k \times 1}$$

By applying chain rule, we obtain:

$$\frac{\partial \mathcal{L}_3}{\partial \mathbf{U}} = \sum_{s}^{S} \sum_{m}^{N_s} \frac{\partial \mathcal{L}_3}{\partial \hat{y}_{sm}} \times \frac{\partial \hat{y}_{sm}}{\partial \mathbf{U}} \tag{4.11}$$

$$\frac{\partial \mathcal{L}_3}{\partial \mathbf{v}_s} = \sum_{m}^{N_s} \frac{\partial \mathcal{L}_3}{\partial \hat{y}_{sm}} \times \frac{\partial \hat{y}_{sm}}{\partial \mathbf{v}_s} \tag{4.12}$$

4.2.4 Algorithm

Equation (4.11) requires us to compute both $G(\hat{y}_{sm}|y_{si}, h^2 + \hat{h}^2)$ and $G(\hat{y}_{sm}|\hat{y}_{si}, 2\hat{h}^2)$, which takes $O(N_s)$. Furthermore, computing the partial derivative of $\mathcal{L}_3$ w.r.t all $y_{sm}, s = 1, 2, ..., S, m = 1, 2, ..., N_s$ requires $O(SN^2)$. To speed up the computation, we employ a mini-batch Gradient Descent (MGD) approach at each iteration, where instead of using all N_s points from each station, we use only a subset of size $l < N_s$. This reduces significantly the amount of computations needed from $O(SN^2)$ to $O(Sl^2)$.

For each gradient descent update step at iteration t, let the mini-batch data matrix be $\mathbf{X}_s^{(t)} \in \mathbb{R}^{l \times d}$ and the mini-batch response vector be $\mathbf{y}_s^{(t)} \in \mathbb{R}^{l \times 1}$. For each station s, we compute the following vector $\mathbf{p}_s \in \mathbb{R}^{l \times 1}$ as the derivative of $\frac{1}{2}\mathcal{L}_3$ on $\hat{y}_s$.

$$\begin{aligned}
& \mathbf{p}_s = \frac{1}{l^2}\left(\frac{\mathbf{G}^s \circ \mathbf{E}^s}{h^2 + \hat{h}^2} - \frac{\mathbf{H}^s \circ \mathbf{F}^s}{2\hat{h}^2}\right) \cdot \mathbb{1} \\
\text{s.t} \quad & \mathbf{G}^s \in \mathfrak{R}^{l \times l} : \mathbf{G}^s_{ij} = G(\hat{y}^{(t)}_{si} | y^{(t)}_{sj}, h^2 + \hat{h}^2) \\
& \mathbf{H}^s \in \mathfrak{R}^{l \times l} : \mathbf{H}^s_{ij} = G(\hat{y}^{(t)}_{si} | \hat{y}^{(t)}_{sj}, 2\hat{h}^2) \\
& \mathbf{E}^s \in \mathfrak{R}^{l \times l} : \mathbf{E}^s_{ij} = \hat{y}^{(t)}_{si} - y^{(t)}_{sj} \\
& \mathbf{F}^s \in \mathfrak{R}^{l \times l} : \mathbf{F}^s_{ij} = \hat{y}^{(t)}_{si} - \hat{y}^{(t)}_{sj} \\
& i, j = 1, 2, ..., l
\end{aligned} \tag{4.13}$$

Thus, the gradient w.r.t $\mathbf{U}$ can be obtained by combining Equations (4.7), ((4.9), and (4.11) as

follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = 2\alpha \mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T + 2\sum_{s}^{S} \mathbf{X}_s^{(t)T}\mathbf{X}_s^{(t)}\mathbf{U}\mathbf{v}_s\mathbf{v}_s^T$$
$$-2\sum_{s}^{S} \mathbf{X}_s^{(t)T}(\mathbf{y}_s^{(t)} - \lambda \mathbf{p}_s)\mathbf{v}_s^T \tag{4.14}$$

Similarly, the gradient w.r.t $\mathbf{V}$ is obtained by combining Equations (4.8), (4.10), and ((4.12) to obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{V}} = 2\alpha \mathbf{U}^T\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A}) + [\Delta \mathbf{a}_1, \Delta \mathbf{a}_2, ..., \Delta \mathbf{a}_S] \tag{4.15}$$

where, $\Delta \mathbf{a}_s = 2\mathbf{U}^T\mathbf{X}_s^{(t)T}\mathbf{X}_s^{(t)}\mathbf{U}\mathbf{v}_s - 2\mathbf{U}^T\mathbf{X}_s^{(t)T}(\mathbf{y}_s^{(t)} - \lambda \mathbf{p}_s)$. Observe that $\mathbf{p}_s$ can be viewed as an adjustment weighted by λ on $\mathbf{y}_s$ when calculating the gradients. In other words, the gradient of the distribution-preserving term $\mathcal{L}_3$ adjusts the role of $\mathbf{y}$ in calculating the gradients. Finally, the Mini-Batch Gradient Descent(MGD) is implemented using the Accelerated Gradient Descent (AGD) approach in order to speed up the search for local optimum [28]. A summary of the algorithm is given in Algorithm 3.

4.3 Experimental Evaluation

We have conducted extensive experiments to evaluate the performance of our proposed framework. The dataset and baseline methods used in our experiments along with the results obtained are described in this section.

4.3.1 Data and Preprocessing

We evaluated the proposed approach on monthly precipitation data from the Global Historical Climatology Network (GHCN) data [85]. The dataset spans a 540-month time period, from January 1970 to December 2014. For brevity, we consider only data from weather stations in the United States (located between $24.74°N$ to $49.35°N$ and $66.95°W$ and $124.97°W$). We also omit any station that has more than 50% missing values in its time series. The resulting dataset contains precipitation data from 1,510 weather stations.

Algorithm 2: DPMTL: Distribution Preserving Multi-task Learning

TRAINING PHASE
Input: Training data $\mathcal{X} = \{\mathbf{X}_s\}$, $\mathcal{Y} = \{\mathbf{y}_s\}$, $\mathbf{A}$, h_s,$\hat{h}_s$, τ_u, τ_v, $s = 1, 2...S$.
Output: $\mathbf{U}$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ...\mathbf{v}_S]$.
repeat
  $t = t + 1$;
  for s = 1,2,...,S
    randomly choose l sample data, $\mathbf{X}_s^{(t)}$ and $\mathbf{y}_s^{(t)}$;
    $\hat{\mathbf{y}}_s^{(t)} = \mathbf{X}_s^{(t)}\mathbf{U}\mathbf{v}_s$;
    compute $\mathbf{p}_s$ with equation (4.13);
  end for
  $\mathbf{U} = \mathbf{U} - \tau_u \frac{\mathcal{L}}{\mathbf{U}}$ by equation (4.14);
  $\mathbf{V} = \mathbf{V} - \tau_v \frac{\mathcal{L}}{\mathbf{V}}$ by equation (4.15);
until converges

PREDICTION PHASE
Input: Testing data $\mathcal{X}^* = \{\mathbf{X}_s^*\}$, $\mathbf{U}$, $\mathbf{V}$.
Output: Predictions $\mathcal{Y}^* = \{\hat{\mathbf{y}}_s^*\}$.
for s = 1,2,...,S
  $\hat{\mathbf{y}}_s^* = \mathbf{X}_s^*\mathbf{U}\mathbf{v}_s$;
end for

We selected 13 predictor variables from the NCEP Reanalysis [56] gridded dataset with the help of our domain expert. A brief introduction of these predictors is shown in Tab. 4.2. The mapping between the GHCN station and its NCEP Reanalysis grid is established by finding the closest grid cell to each GHCN station. Variables of each station is deseasonalized by subtracting the seasonal mean of that station and dividing by the corresponding seasonal standard deviation.

4.3.2 Experimental Setup

We compared the proposed framework, DPMTL, against the following baseline algorithms:

- **Global model**: The data from all stations are combined and used to train a global, lasso regression model.
- **Local model**: A local model is trained for each station using only data from the given station.

Table 4.2: Predictor variables from NCEP reanalysis.

Variable	Description
cprat	convective precipitation rate at surface
dlwrf	longwave radiation flux at surface
dswrf	solar radiation flux at surface
lftx	surface lifted index
omega	omega at sigma level 0.995
pr_wtr	precipitable water content
prate	precipitation rate
rhum	relative humidity at sigma level 0.995
slp	Sea level pressure
thick850	thickness for 850-500mb
thick1000	thickness for 1000-500mb
tmax	maximum temperature at 2 m

- **GSpartan** [136]: A MTL framework for spatio-temporal data. This framework is equivalent to setting the hyperparameter λ in DPMTL to 0 and adding L_1 regularizers to the model parameters $\mathbf{U}$ and $\mathbf{V}$.

- **Contour Regression**: A distribution-preserving method for time series prediction [5]. Unlike DPMTL, contour regression is designed to improve distribution fit by minimizing the discrepancy between the empirical cumulative density function of the predicted and ground truth values, whereas DPMTL applies L^2-distance on probability density functions estimated using KDE. Furthermore, contour regression is a single-task learning method, unlike the multi-task learning method used in DPMTL.

The evaluation metrics used in this study are defined below, where $\hat{\mathbf{y}}_s$ corresponds to the predicted values for station s and $\mathbf{y}_s$ corresponds to their true values.

- **RMSE**: a measure of prediction error obtained by taking the square root of the average sum-of-squared errors in the predictions.

$$\text{RMSE} = \frac{1}{S}\sum_{s}^{S}\sqrt{\frac{1}{N_s}\sum_{i}^{N_s}(y_{si} - \hat{y}_{si})^2}$$

- **RMS-CDF**: a metric defined in [5] to evaluate the fit between two cumulative distribution functions created from a finite sample of observations. The metric is equivalent to applying RMSE on the ordered values of the data.

$$\text{RMS-CDF} = \frac{1}{S}\sum_{s}^{S}\sqrt{\frac{1}{N_s}\sum_{i}^{N_s}(y_{s(i)} - \hat{y}_{s(i)})^2}$$

- L^2**-Distance**: another metric for measuring the divergence between two density functions, computed according to the formula given in Theorem 1.

$$\begin{aligned} & L^2\text{-Distance} \\ = \; & \frac{1}{S}\sum_{s}^{S}\frac{1}{N_s^2}\sum_{i,j}^{N_s}\left(G(y_i|y_j, 2h^2) + G(\hat{y}_i|\hat{y}_j, 2\hat{h}^2) - 2G(y_i|\hat{y}_j, h^2 + \hat{h}^2)\right) \end{aligned}$$

We apply 9-fold cross validation on the 45-year data (from 1970-2014) to evaluate the performance of various algorithms. Each fold corresponds to 5 years or 60 months worth of data. In each of the 9 rounds, 8 of the folds are selected to be the training set while the remaining fold is used as test set. Since the dataset has been standardized by their corresponding month, we set the Parzen window width h_s and $\hat{h}_s$ to be half of its variance, i.e., 0.5. The spatial autocorrelation matrix $\mathbf{A} = \exp(-\mathbf{Q}/\gamma)$ is computed using the Haversine distance $\mathbf{Q}$, with $\gamma = 100$. The number of latent factors k is set to 10 while the mini-batch size l is chosen to be 64. For gradient descent, the step sizes τ_u and τ_v are initialized to 10^{-8} and 10^{-7} and gradually decreased with increasing number of iterations.

The hyperparameters α and λ are tuned via nested cross-validation on the training data. Since we want to minimize both residual and distribution errors, their trade-off must be considered during hyperparameter tunning. Let RMSSUM= $(1-\beta)$RMSE + β RMS-CDF, where β is a parameter that controls the tradeoff between minimizing RMSE and RMS-CDF. The hyperparameters for all competing algorithms are chosen in such a way to minimize the RMSSUM on the validation set. To ensure fair comparison, we report the performance of all the algorithms for each β chosen in the range between [0,1]. For example, if $\beta = 0$, the chosen hyperparameters will be biased toward minimizing RMSE.

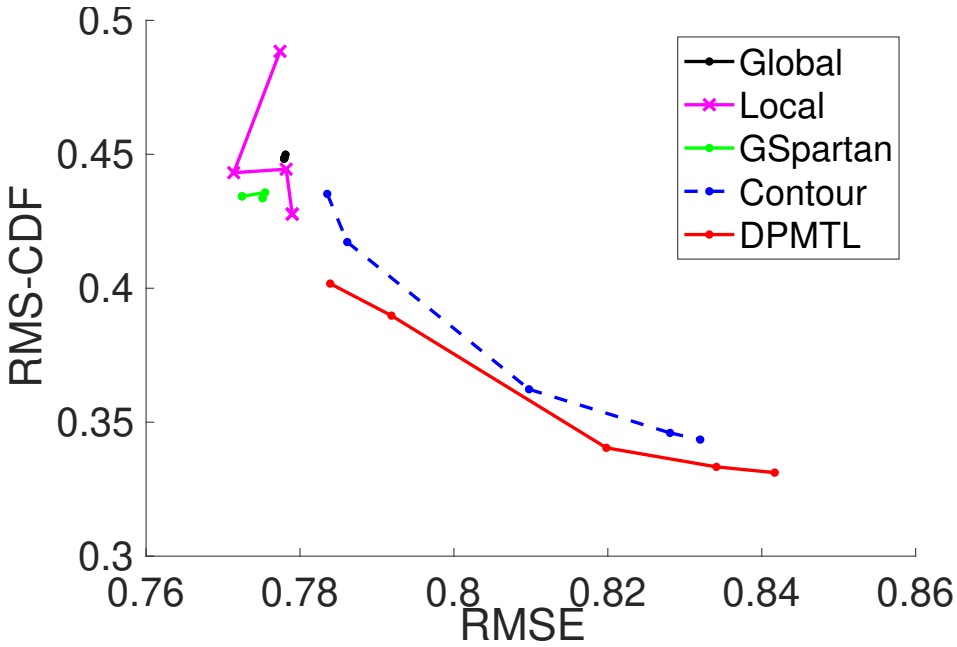


Figure 4.2: Performance comparison between DPMTL and baseline approaches in terms of RMSE and RMS-CDF when varying the tradeoff parameter β between 0 and 1.

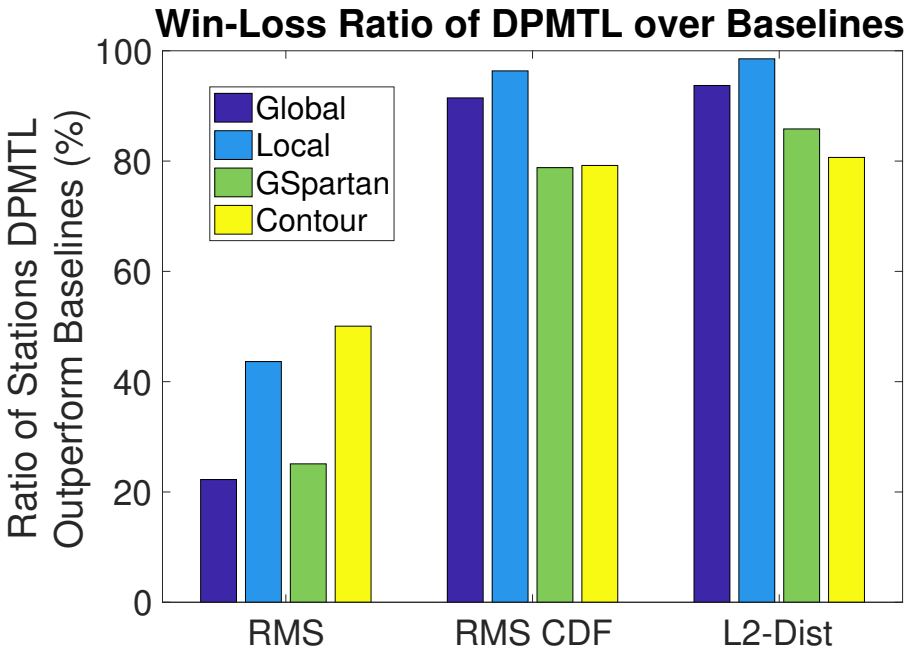


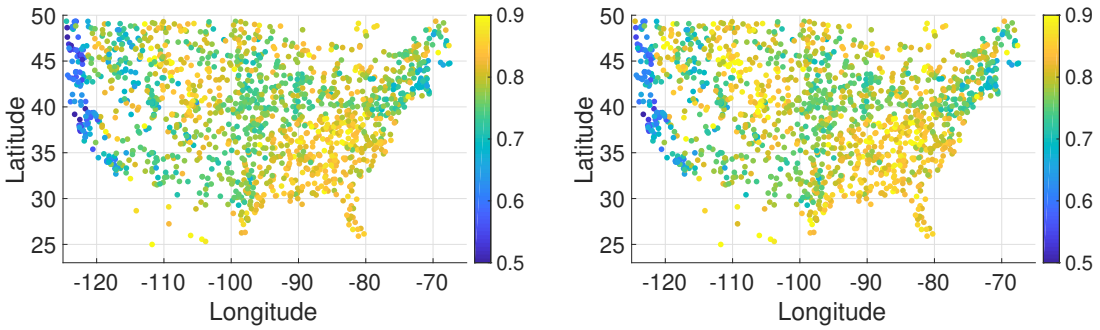
Figure 4.3: Percentage of stations in which DPMTL outperforms the baseline methods (for $\beta = 0$).

4.3.3 Experimental Results

Figure 4.2 summarizes the average performance of the various algorithms after 9-fold nested cross validation. For each algorithm, their RMSE and RMS-CDF metrics are computed as β is varied from 0, 0.25, 0.5, 0.75 to 1. The results suggest that Global (lasso) and GSpartan are not sensitive to the tradeoff parameter β, which is obvious since they are both non-distribution preserving methods. For Local (lasso) models, varying β reduces RMS-CDF slightly, though the change is more erratic as the tuned hyperparameters are highly sensitive to the training set size. For distribution preserving methods such as contour regression and DPMTL, Figure 4.2 suggests that there is a trade-off between minimizing prediction error and distribution error. Increasing β monotonically reduces RMSE-CDF at the expense of increasing RMSE. When the hyperparameters are tuned to minimize RMS-CDF ($\beta = 1$), DPMTL has the lowest RMS-CDF with only a slight increase in RMSE compared to the non-distribution preserving methods. Specifically, the increase in RMSE is only between $0.75\% - 1.48\%$ compared to the significant reduction in RMS-CDF between $7.5\% - 17.75\%$. DPMTL also outperforms contour regression, a distribution-preserving single-task learning method. In fact, the curve for DPMTL is lower than that for contour regression, which justifies our rationale for developing a distribution-preserving MTL framework. Furthermore, the improvement in RMS-CDF for DPMTL is more pronounced for larger values of β. For example, when $\beta = 0.5$, DPMTL achieves a reduction in RMS-CDF by more than 21.9% compared to global, local, and GSpartan, with an increase in RMSE by at most 5.72%.

We also analyze the performance of the algorithms on a station by station basis. Figure 4.3 shows the percentage of stations in which DPMTL outperforms each baseline method according to the given metrics. The results show that DPMTL outperforms both Global and Local models in more than 91% of the stations. It also outperforms GSpartan and Contour in more than 78% of the stations (in terms of RMS-CDF) and in more than 80% of the stations (in terms of L^2-Distance).

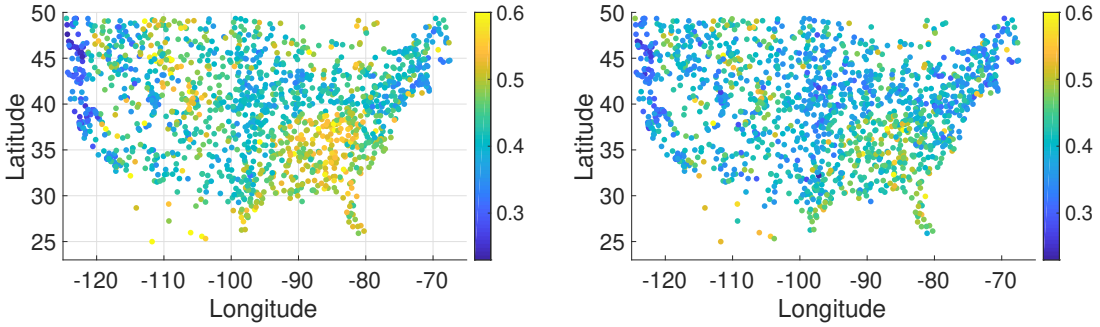
Figure 4.4 compares the RMSE values of GSpartan and DPTML for all stations in the dataset. While the overall RMSE for DPMTL is slightly worse than GSpartan, the maps shown in Figure 4.4 are quite similar to each other. The poor performance of DPMTL tends to occur at locations



(a) RMSE results for GSpartan.

(b) RMSE results for DPMTL.

Figure 4.4: Comparison between the RMSE of GSpartan and DPMTL for $\beta = 0$ (figure best viewed in color).



(a) RMS-CDF error for GSpartan.

(b) RMS-CDF error for DPMTL.

Figure 4.5: Comparison between the RMS-CDF of GSpartan and DPMTL for $\beta = 0$ (figure best viewed in color).

where GSpartan also has high RMSE. However, in terms of their RMS-CDF, the maps shown in Figure 4.5 suggest that the distribution fit of DPMTL improves significantly for the majority of the stations. GSpartan performs poorly with high prediction and distribution errors especially in the southeastern part of the United States, where there are more variability in their precipitation time series. Although the RMSE is also high for DPMTL in this region, its RMS-CDF improves significantly. The maps also show that the distribution error is generally lower for both methods along the Pacific and Atlantic coastal areas.

Finally, we also examine characteristics of the predicted distribution generated by different algorithms. Figure 4.6 shows the precipitation histograms obtained using DPMTL, contour regression, and GSpartan for a station located at [38.25°N, 82.99°W]. The results suggest that DPMTL

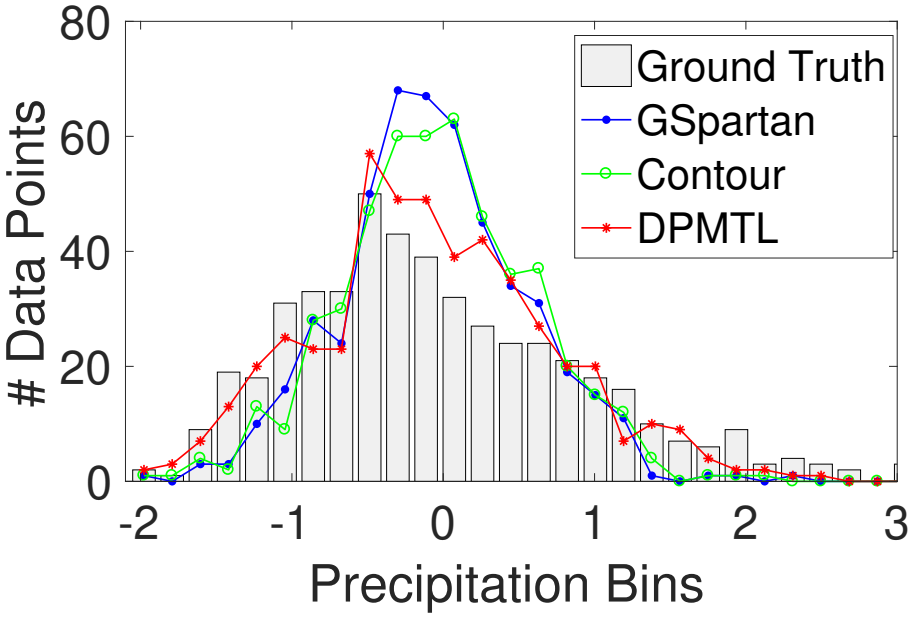


Figure 4.6: Histogram comparison of precipitation distribution for a weather station located at [38.25°N, 82.99°W].

has the best fit to the ground truth distribution compared to contour regression and GSpartan. In particular, DPMTL was able to capture the skewness and heavy tail distribution. DPMTL also fits the distribution of below average precipitation values more effectively than the other two methods. Although the plot was shown only for one station, the good fit obtained by DPTML was found in many other stations in our dataset.

4.4 Related Work

Multi-task learning (MTL) is a machine learning technique for solving multiple related modeling tasks jointly by exploiting and sharing the common information among tasks [148]. MTL improves generalization performance by leveraging domain-specific information to enable the pooling of information across different tasks, which is particularly useful when there are insufficient training instances available to solve each prediction task separately. It also provides a natural way to handle multi-location prediction problems [147] [137] [136]. For example, in [136], the prediction at each location can be considered a single task, which is related to the prediction tasks at other nearby locations. However, existing MTL methods focus primarily on minimizing point-wise prediction

errors, ignoring how well the predicted distribution fits the true distribution of the data. Alternative approaches such as quantile mapping [82] have been developed to correct the bias between the true and predicted distribution. However, such techniques tend to have poor prediction accuracy [5]. While hybrid approaches such as contour regression has been developed [5], they are mostly designed for single-task learning.

4.5 Summary

This chapter presents a distribution preserving multi-task regression framework for spatio-temporal data. Our framework employs a Parzen window based kernel density estimation (KDE) approach to compute the probability density function and L^2-distance to measure the difference between two distributions. We evaluated our method on a real-world climate dataset, containing more than 1500 stations in the United States, and showed that the proposed framework outperforms four other competing baselines in at least 78% of the stations.

CHAPTER 5

MULTI-TASK HIERARCHICAL LSTM FRAMEWORK FOR MULTI-STEP-AHEAD TIME SERIES FORECASTING

This chapter investigates the challenge of solving multi-step-ahead time series forecasting using a multi-task learning approach. There are two key differences between the framework proposed in this chapter and the one developed in the previous chapter. First, instead of making single-step prediction, the framework proposed here is designed to generate forecasts at multiple consecutive future time steps. Second, the proposed framework is nonlinear, using a hierarchical long short-term memory (LSTM) architecture to capture the temporal dependencies of the data. As proof of concept, the proposed framework will be applied to the problem of forecasting monthly sea surface temperature using a suite of ensemble member forecasts from dynamical (physical) models as its predictor variables.

The remainder of this chapter is organized as follows. The motivation and challenges of predicting sea surface temperature are described in Section 5.1, followed by a review of related literature in Section 5.2. The formal problem statement and background on LSTM architecture are given in Section 5.3. The proposed hierarchical LSTM structure is then described in Section 5.4. Section 5.5 presents the experimental results followed by conclusions in Section 5.6.

5.1 Ensemble Forecasting of Sea Surface Temperature

With more than two-third of the Earth covered by ocean, accurate prediction of sea surface temperature (SST) is crucial due to its significant influence on the climate patterns around the world. For example, the El Niño phenomenon, which is related to the abnormally high sea surface temperature values in the equatorial Pacific Ocean, has been shown to cause unusual droughts and extreme rainfall across many regions [49]. In addition, SST is a key parameter for weather prediction and atmospheric model simulations and contributes to the development of tropical cyclones such as hurricanes [30]. Thus, its accurate prediction is essential and remains an active

research area [52, 78, 48].

Various dynamic forecasting systems, such as NCEP coupled forecast system model (CFSv2) and the Canadian Seasonal to Interannual Prediction System (CanCM3), have been developed over the years to predict SST and other climate variables. These models would simulate the physical processes that govern the fundamental dynamics of the Earth system. However, predicting the future states of a such a highly nonlinear dynamic system is very challenging, even with these state-of-the-art models. In particular, the predictions made by these models will likely never be perfect due to the uncertainties associated with the models themselves, their initial conditions, and the chaotic nature of the ocean and climate system. While the errors in predictions may be acceptable at for short-term predictions, these errors can compound and accumulate as predictions are made for more distant times. To better represent the effect of initial condition uncertainties, ensemble prediction with perturbed initial conditions has been adopted in operational forecasting [26]. To represent the uncertainties associated with models such as model physics, parameterization schemes and resolution, multi-model ensemble (also called superensemble) approach was introduced for SST predictions. Instead of a single forecast run from a single model, ensemble forecast from a group of models, each producing an ensemble of forecast from different initial conditions, are used to better estimate forecast uncertainties. The multi-model ensemble approach has been demonstrated to provide more reliable forecast than any single model.

The North American Multi-Model Ensemble (NMME) [62] is an example of a multi-model ensemble for climate prediction, including monthly SST. Fig. 1.3 shows an example of the monthly SST predictions from NMME for the time period between June 2010 and November 2010. The time at which the forecasts were generated is known as *forecast generation time* whereas the number of months ahead the forecast was made is called the *forecast horizon* or *lead time*. For example, the forecasts shown in Fig. 1.3 are for lead times up to 8 months. Each month, every physical model is run multiple times with slightly perturbed initial conditions to create several different *instances* of each model. Each subplot of Figure 1.3 contains the entire ensemble of all models and all of their instances (i.e. the superensemble) generated within a single month.

However, in many cases it is preferable to have a single point estimate of the expected SST rather than a superensemble of disparate SST predictions. With multi-model ensemble prediction, a common way to derive a deterministic forecast from the superensemble is to take the average or median [103], while other methods have been developed to weigh the models differently based on their performance [135]. These previous approaches suffer from two main deficiencies. First, they are unable to capture complicated non-linear relationships between different ensemble members. Second, they may not fully capture the temporal autocorrelation present in the time series. The latter problem is illustrated in Fig. 5.1. Each row in the diagram corresponds to a set of predictions generated at a given forecast generation time while each column represents the set of forecasts generated for a given lead time. The figure shows there are two types of temporal dependencies that should be modeled in multi-step-ahead time series prediction:

1. Temporal autocorrelation between different lead time forecasts for the same forecast generation time. This is denoted as lead time level autocorrelation (i.e., autocorrelation between elements in each row) in Fig. 5.1.

2. Temporal autocorrelation of all forecasts for a given lead time. This is denoted as generation time level autocorrelation (i.e., autocorrelation between elements in each column) in Fig. 5.1.

Incorporating both types of autocorrelation into the learning framework would be useful to ensure robustness of the models and temporal consistencies of their predictions especially for noisy data.

To address these challenges, this chapter presents a novel hierarchical LSTM framework for aggregating the multi-model ensemble forecasts of SST. The proposed architecture consists of three main components. The first component is a lead time encoder LSTM for extracting a high-level representation of the predictions made at each lead time. The second component is the generation time encoder which is an LSTM that extracts a high-level representation of the physical model predictions for a particular lead time and generation time window. The third component is a fully connected network to convert the output of the generation time encoder to a final prediction.

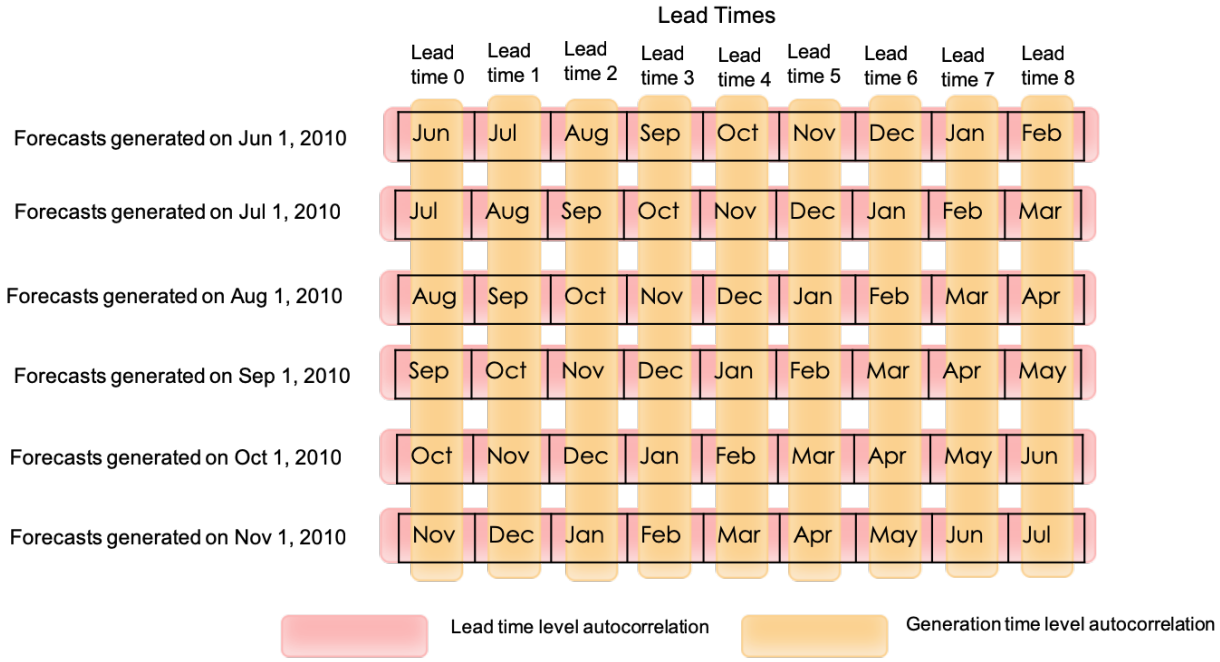


Figure 5.1: A two-level autocorrelation structure in multi-lead time forecasting of the SST data shown in Fig. 1.3.

5.2 Related Work

To address the multi-step-ahead time series forecasting problem, a novel MTL framework using hierarchical LSTM is proposed in this chapter. This section reviews previous research on multi-task learning in deep neural networks (DNN) and hierarchical LSTM.

5.2.1 Multi-task Learning in DNN

Deep learning has recently exploded in popularity thanks to its ability to effectively model complicated non-linear relationships. It has been succesfully applied to images [64], video [112], and natural language processing [124] among other important tasks. The conventional deep learning approaches to multi-task learning can be categorized into two types: hard parameter sharing and soft parameter sharing of hidden layers [105]. In terms of hard parameter sharing, multiple neural networks are built, and they are constrained to share the parameters at a certain number of layers while learning parameters on their own for the rest layers [21]. In [53], the cross-language layer

is shared by the model of all languages, and differentiates only at the output layer. In terms of soft parameter sharing, regularizer skills are employed to constrain the correlations between model parameters from different tasks [37, 143].

5.2.2 Hierarchical LSTM

Long short-term memory (LSTM) network is one of the most popular deep learning architecture for modeling sequential data such as time series, where the data points exhibit strong temporal autocorrelation, and document data, where the appearance of a word depends highly on its context. Hierarchical LSTM [144, 44, 100] is a variant of LSTM that has been developed in recent years to capture the myriad types of relationships and processing of sequential data.

For example, in document modeling, a hierarchical LSTM architecture was proposed in [144] to represent the multi-level structure of a document. Specifically, a document contains words that are sequentially connected to form a sentence, which in turn, is connected to other sentences to form a paragraph. In [144], the first level of the LSTM hierarchy encodes the word level dependencies, while the second level encodes the sentence level dependencies. The document-sentence-word hierarchy of the data is analogous to the two-level temporal correlation shown in Fig 5.1. A hierarchical LSTM framework was also developed in [44] for the multi-modal data fusion problems, where each modality may refer to sensors, video, audio, etc. The first LSTM layer of the hierarchy learns the modality-specific temporal dynamics whereas the second layer combines the representation of each modality to generate an embedding for each time step. However, none of these hierarchical LSTM architectures [144, 44] are designed for multi-task learning, and thus, cannot be easily adapted to the multi-step-ahead time series prediction problem.

In [100], a hierarchical LSTM with attention mechanism was developed for time series prediction with multiple input (driving) time series. The proposed architecture builds upon previous research on attention mechanism [13] to improve performance of RNN. Specifically, the attention mechanism acts as a selection/filtering mechanism to the input data. The first layer of their architecture uses an input attention mechanism to extract the relevant input variables for the analysis while the second

layer uses an LSTM with temporal attention mechanism to learn a hidden representation for each time step and selects the relevant hidden states for subsequent time series prediction task. The outputs of the second layer are then fed into another LSTM for modeling the response time series. Unfortunately, the original architecture was not designed for multi-task learning problems. It has to be modified to make predictions at multiple future time steps, without using the ground truth values of the shorter-term forecasts to make longer-term forecasts. A modified version of this architecture is therefore used as one of the baseline methods in the experiment section.

5.3 Preliminaries

This section formalizes the multi-step time series forecasting problem and introduces the basic formulation of LSTM model.

5.3.1 Problem Statement

Let $\mathbf{x}_{<t,l>} \in \mathbb{R}^M$ be an input vector of predictor variables generated at time t for the forecast at lead time l, where $t \in [1, T]$ and $l \in [1, L]$. Furthermore, let $y_{t+l} \in \mathbb{R}$ be the corresponding ground truth value of the target variable at time $t + l$. The multi-step-ahead time series forecasting is formally defined as follows.

Definition 1 (Multi-Step-Ahead Time Series Forecasting) *Given a training set, $\mathcal{D} = \{\mathbf{X}_t, \mathbf{y}_t\}_{t=1}^T$, where $\mathbf{X}_t \in \mathbb{R}^{L \times M}$ and $\mathbf{y}_t \in \mathbb{R}^L$, multi-step-ahead time series forecasting seeks to learn a target function $f_l : \mathbb{R}^M \rightarrow \mathbb{R}$ that maps each input vector $\mathbf{x}_{<t,l>} \in \mathbb{R}^M$ to its corresponding output y_{t+l}.*

Let $o_{<t,l>} \equiv f_l(\mathbf{x}_{<t,l>})$ denote the output of the target function for lead time l when applied to the predictor variables generated at time t. The effectiveness of the target function output can be measured by comparing it against the ground truth value y_{t+l}.

In the context of ensemble forecasting of SST, the predictor variables correspond to forecasts generated by a set of M physical models. Specifically, at each generation time t, a physical model m is run to generate forecasts up to L future time steps (lead times). Each physical model is run by d_m

times, with a varying initial and boundary conditions. The output of each d_m run is also known as an ensemble member forecast and d_m is the number of ensemble members associated with model m. The forecasts of these d_m ensemble members are often averaged together into a single point prediction for model m. The resulting averaged forecasts of the M models are then used as input to the proposed framework.

5.3.2 Long Short-Term Memory (LSTM) Network

Our model is based upon the popular Long Short-Term Memory network (LSTM), a type of Recurrent Neural Network (RNN) [50, 46] that has proven to be effective in dealing with sequential data, especially time series data with long-term dependencies. At each time step, i, the hidden state of the LSTM, $\mathbf{h}_i$, depends on the hidden state from the previous time step, $\mathbf{h}_{i-1}$, as well as the input in the current time step, $\mathbf{x}_i$. We compactly represent this relationship using the following equation:

$$\mathbf{h}_t = \text{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}) \tag{5.1}$$

LSTM maintains the long-term history of a time series with a cell state $\mathbf{c}_t$. The information that flows into and out of the cell state are regulated with several gates. First, the input $\mathbf{x}_t$ at current time t and the previous hidden state $\mathbf{h}_{t-1}$ are used to change the cell memory as follows:

$$\bar{\mathbf{c}}_\mathbf{t} = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)$$

The amount of change to the current cell state is controlled by an input gate $\mathbf{i}_t$ while the amount for the cell to maintain its previous state information is regulated by a forget gate $\mathbf{f}_t$:

$$\mathbf{c}_t = \mathbf{i}_t \circ \mathbf{c}_{t-1} + \mathbf{f}_t \circ \bar{\mathbf{c}}_\mathbf{t},$$

where the input and forget gates also depend on $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$.

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) \\ \mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f) \end{aligned}$$

The cell state provides the information needed to compute the output $\mathbf{h}_t$, whose value is regulated by an output gate.

$$\begin{aligned} \mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t) \\ \mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \end{aligned}$$

In the above equations, $\sigma(\cdot)$ denotes a sigmoid activation function, $\tanh(\cdot)$ denotes a hyperbolic tangent function, and $\circ$ denotes the Hadamard product operation.

5.4 Proposed MSH-LSTM Architecture

This section presents an overview of the proposed multi-step-ahead hierarchical LSTM (MSH-LSTM) framework for time series forecasting. The objective of MSH-LSTM is to jointly train a set of inter-dependent LSTM models that capture both the temporal dependencies between lead times as well as those between consecutive generation times, as illustrated in Fig. 5.1. A schematic diagram of the proposed MSH-LSTM architecture is shown in Fig.5.2. The architecture is composed of three layers—a lead time encoder layer, a generation time encoder layer, and an output layer. Details of each layer are discussed next.

5.4.1 Lead Time Encoder Layer

This layer enforces the temporal dependencies between different lead time forecasts for the same forecast generation time. Specifically, let $\mathbf{v}_t = \{\mathbf{x}_{<t,1>}\mathbf{x}_{<t,2>} \cdots \mathbf{x}_{<t,L>}\}$ be a sequence of length L corresponding to the ensemble forecasts generated at time t for each lead time $l \in [1, L]$, where each element of the sequence is an M-dimensional vector, i.e., $\mathbf{x}_{<t,l>} \in \mathbb{R}^M$. The lead time encoder layer takes this sequence as input and produces a sequence of hidden states $\{\mathbf{h}_{<t,1>}\mathbf{h}_{<t,2>} \cdots \mathbf{h}_{<t,L>}\}$ of the same length using an LSTM network, LSTM^{lead}, where each $\mathbf{h}_{<t,l>} \in \mathbb{R}^d$. The outputs of LSTM^{lead} can be viewed as feature representation for each forecast lead time l, embedded with the temporal dependencies between them. More formally, the relationship between the input and

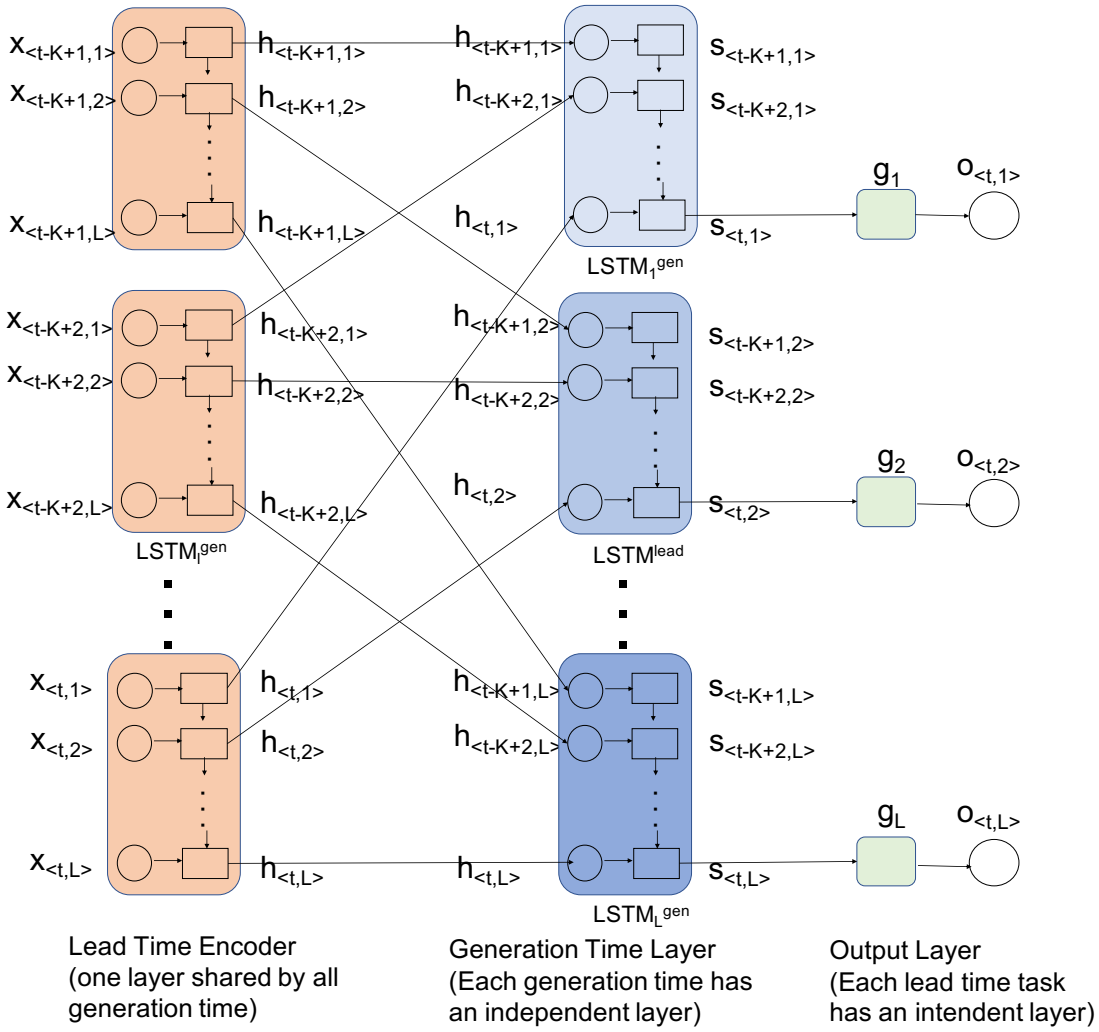


Figure 5.2: Proposed hierarchical LSTM architecture for multi-step-ahead time series forecasting. Blocks with different shades of colors (in the generation time and output layers) are trained independently and have different parameters while those with the same color (lead time encoder layer) are trained jointly and have identical parameters.

output of the lead time encoder can be expressed as follows:

$$\mathbf{h}_{<t,l>} \quad = \quad \text{LSTM}^{lead}(\mathbf{h}_{<t,l-1>}, \mathbf{x}_{<t,l>}) \tag{5.2}$$

The structure for LSTM^{lead} is depicted by the orange boxes at the first (left-most) layer of Fig. 5.2. Each box is assumed to process a sequence of length L from different generation times, i.e., $\mathbf{v}_{t-K+1}, \mathbf{v}_{t-K+2}, \cdots, \mathbf{v}_t$. Although they are depicted as separate boxes, note that the parameters of LSTM^{lead} are shared across the boxes, i.e., the parameter values are identical when processing every sequence in the time window K.

5.4.2 Generation Time Encoder Layer

The second layer of MSH-LSTM, which corresponds to the generation time encoder, models the temporal dependencies between forecasts generated at different times in a given time window, K, for each lead time l. To illustrate this, let t be the current time step and $\mathbf{w}_l = \{\mathbf{h}_{<t-K+1,l>}\mathbf{h}_{<t-K+2,l>} \cdots \mathbf{h}_{<t,l>}\}$ be a sequence of length K, whose elements correspond to the outputs of the hidden states generated for the same lead time l by the previous layer of MSH-LSTM. In the case when $t < K$, zero padding is used to ensure every sequence $\mathbf{w}_l$ is of the same length, K, at all times.

The generation time encoder will take the sequence for a specific lead time l (i.e., w_l) as input and produces another sequence of hidden states $\mathbf{s}_{<t-K+1,l>}\mathbf{s}_{<t-K+2,l>} \cdots \mathbf{s}_{<t,l>}$ as output using an LSTM network LSTM_l^{gen}, where each $\mathbf{s}_{<t,l>} \in \mathbb{R}^p$ and $l \in [l, L]$. More specifically, the relationship between the inputs and outputs of the generation time encoder for lead time l can be expressed as follows:

$$\mathbf{s}_{<t,l>} = \text{LSTM}_l^{gen}(\mathbf{s}_{<t-1,l>}, \mathbf{h}_{<t,l>}) \tag{5.3}$$

The outputs of LSTM_l^{gen} can be viewed as a representation embedding by taking into account the temporal dependencies between different forecast generation times (within the time window K) for a given lead time l. Furthermore, note that $\mathbf{s}_{<t,l>}$ incorporates information about both the lead time level autocorrelation and generation time level autocorrelation shown in Fig. 5.1. In details, while $\mathbf{s}_{<t,l>}$ explicitly learns the temporal dependencies between different forecast generation times using Eq. (5.3), and the temporal dependencies between the lead times are implicitly captured through the hidden states $\mathbf{h}_{<t,l>}$ as its input.

Unlike the lead time encoder layer, the parameters of the generation time encoders are not shared, because the temporal relationships vary for different lead times. Such varying parameter values are illustrated by the different shades of blue boxes in Fig. 5.2, where each box is assumed to process a sequence of length K for different lead times, i.e., $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_L$.

5.4.3 Output Layer

Finally, the output layer of MSH-LSTM will take each hidden state $\mathbf{s}_{<t,l>}$ as input and uses a fully connected network to generate its prediction for lead time l at generation time t:

$$o_{<t,l>} = g_l(\mathbf{s}_{<t,l>}) \tag{5.4}$$

The fully connected networks are depicted by the green boxes in Fig. 5.2. As we assume that the models may vary across different lead times, the parameters are not shared across different lead times, and an independent g_l is learned for each lead time. This is depicted by the varying shades of green boxes in the diagram.

5.4.4 Parameter Estimation

Let Θ be the set of parameters associated with the proposed MSH-LSTM framework to be estimated from data. MSH-LSTM is a multi-task learning framework as the model parameters for all lead times are (1) tied via the hierarchical LSTM structure and (2) jointly estimated by optimizing the following least-square loss function:

$$\Theta^* = \arg\min_{\Theta} \sum_{t}^{T} \sum_{l}^{L} (o_{<t,l>} - y_{t+l})^2 \tag{5.5}$$

The network parameters in Θ are initialized randomly and then trained in an end-to-end fashion using Adam [61]. To avoid overfitting, a dropout strategy is employed during the training process, which improves the robustness of the network. The stopping criteria for training the network depends on its performance on a separate validation set. Since the error on validation set generally has a decreasing trend as the training epochs iterate, the training process is terminated when the validation error converges. A pseudocode summarizing the training procedure is shown in Algorithm 3. Hyperparameters of the framework such as the number of nodes at each layer, learning rate, batch size, and dropout rate are also tuned based on the performance of the framework on validation set. The entire architecture was implemented in *PyTorch* [95].

Algorithm 3: Training process for MSH-LSTM: Multi-task Hierarchical LSTM

Input: Training set $\mathcal{D} = \{\mathbf{X}_t, \mathbf{y}_t\}_{t-1}^{T}$, K, and dropout rate p;
Output: Parameter set Θ^* for LSTM^{lead}, LSTM_l^{gen}, and g_l (where $l \in [1, L]$);
$i = 0$;
Initialize Θ randomly as $\Theta^{(0)}$;
repeat
  for batch = 1,2,...
    $i = i + 1$;
    Randomly drop out neural network units with rate p;
    Update $\Theta^{(i)}$ with back propagation;
    Recover the network units that were dropped out;
  end for
  Compute error on validation set by replacing $\Theta^{(i)}$ into Equations (5.2), (5.3), (5.4), (5.5);
until validation error converges
$\Theta^* = \Theta^{(i)}$

5.5 Experimental Results

5.5.1 Data Set

The performance of the proposed MSH-LSTM framework is evaluated using an ensemble of monthly sea surface temperature forecasts from the North American Multi-Model Ensemble (NMME) project [62]. Monthly SST observations are collected for a 10-year period from January 1982 to December 2010 for a total of 384 months. The forecast lead times are set to a maximum of 9 months, starting from the end of the current month[1] to the end of 8 months ahead, for a total of $L = 9$ prediction tasks. The data was obtained from 58 grid cells located in the tropical Pacific area. Forecasts from $M = 7$ physical models, which are listed in Table 5.1, are used as predictor variables. As has been mentioned before, although each physical model generates multiple ensemble member forecasts (from varying initial conditions), the average forecast value of the members was used to represent the predictions of each physical model. The window size K for creating sequences of different forecast generation times is set to 6.

The effectiveness of the proposed framework was evaluated on 6 different training-validation-

[1] As the forecasts are generated at the beginning of the month, forecasting the current month means predicting the average sea surface temperature over the course of the month that is currently in progress.

Table 5.1: Physical models from NMME used for monthly sea surface temperature prediction

Index	Model name	# ensemble members
1	CMC1-CanCM3	10
2	CMC2-CanCM4	10
3	COLA-RSMAS-CCSM3	6
4	COLA-RSMAS-CCSM4	10
5	GFDL-CM2p1	10
6	NCEP-CFSv2	10
7	NCAR-CESM1	24

Table 5.2: Partitioning of SST data into multiple training, validation, and test splits.

Data Set	Training Period	Validation Period	Testing Period
SST1	Jan 1982 - Apr 2006	Jan 2007 - Aug 2008	May 2009 - Dec 2010
SST2	Jan 1982 - Aug 2004, Jan 2010 - Dec 2010	May 2005 - Dec 2006	Sep 2007 - Apr 2009
SST3	Jan 1982 - Dec 2002, May 2008 - Dec 2010	Sep 2003 - Apr 2005	Jan 2006 - Aug 2007
SST4	Jan 1982 - Apr 2001, Sep, 2006 - Dec 2010	Jan 2002 - Aug 2003	May 2004 - Dec 2005
SST5	Jan 1982 - Aug 1999, Jan 2005 - Dec 2010	May 2000 - Dec 2001	Sep 2002 - Apr 2004
SST6	Jan 1982 - Dec 1997, May 2003 - Dec 2010	Sep 1998 - Apr 2000	Jan 2001 - Aug 2002

testing splits, as shown in Table 5.2. To avoid overlap due to the multi-step-ahead predictions, a gap of 9 months is introduced between the training-validation and validation-testing sets. Each split has 20 generation time steps for validation and another 20 generation time steps for testing. Since the data is collected over 58 grid cells, there are altogether 20 (generation time steps) $\times$ 58 (grid cells) $\times$ 9 (lead times) test instances to be predicted in each split. Furthermore, the sequences in the training, validation, and testing sets are centered by subtracting their monthly values with the corresponding monthly means computed from the training set.

5.5.2 Baseline Algorithms

The proposed MSH-LSTM framework was compared against the following baselines:

- **EnS**: This is a simple approach that uses the ensemble mean to form a point estimate for the ensemble of model forecasts [26].

- **Ridge**: In this approach, an independent ridge regression model is trained for each lead time task. The input of the model is a 7-dimensional vector, corresponding to the averaged ensemble member forecasts for each of the 7 physical models.

- **FFN**: This approach trains an independent feed-forward neural network with two hidden layers and an output layer for each lead time. Hyperparameters to be tuned on validation set include the number of nodes in each layer, dropout rate, learning rate and batch size. The input of the model is similar to that for ridge regression.

- **LSTM**: An independent LSTM model is trained for each lead time task. The input to the LSTM is slightly different from that for ridge regression and feed-forward neural network as it requires a sequence of length 6 ($K = 6$), where each element of the sequence is a 7-dimensional vector.

- **GFFN**: This global approach trains a single feed-forward neural network to predict all 9 lead time tasks. The input is similar to that for ridge regression and feed-forward neural network.

- **GLSTM**: This is similar to the previous global approach except it uses LSTM instead of a feed-forward network. The input of the model is similar to that for independent LSTM models.

- **ARIMA**: This corresponds to the autoregressive integrated moving average model, which is typically used for time series prediction [81]. For each lead time task, an independent ARIMA model is trained. Its input corresponds to historical SST values up to 6 previous time steps, i.e., it does not use model forecasts from NMME.

- **DARNN**: This is a state-of-the-art hierarchical LSTM network for time series prediction with exogenous variables [100]. It is based on a dual stage attention-based recurrent neural network model. Although it is a hierarchical LSTM, its multi-level structure is designed to capture relationships between its input features (i.e., physical model forecasts) and forecast generation times, but not the dependencies between different lead times, unlike the proposed

MSH-LSTM framework. Since the network is not designed for multi-step-ahead prediction, an independent DARNN model is trained for each lead time task. The input to the model is a sequence of length 6, similar to that for LSTM and MSH-LSTM.

- **MTL-LSTM**: A 3-layer LSTM network based on the multi-task learning approach described in [105, 21]. The bottom layer is a feed forward layer shared by all lead time tasks. The middle layer is a standard LSTM while the top (output) layer is a feed forward neural network. While the model parameters at the bottom layer are shared, both the middle and top layers are built independently for each lead time task. Its input corresponds to the ensemble model forecasts for all 9 lead times with a time window of 6 forecast generation times. In other words, the input is a length-6 sequence of 7×9-dimensional matrices.

5.5.3 Evaluation Metric

The prediction error for each method is computed using the root mean square error (RMSE) metric. The metric can be computed independently for each lead time prediction task as well as for all the tasks:

$$\text{Error for a given lead time, } l\text{: } RMSE_l = \sqrt{\frac{1}{T}\sum_{t}^{T}(y_{t+l} - o_{<t,l>})^2}$$

$$\text{Overall error for all lead times: } RMSE = \sqrt{\frac{1}{T}\sum_{t}^{T}\sum_{l}^{L}(y_{t+l} - o_{<t,l>})^2}$$

where $o_{<t,l>}$ denotes the predicted value for lead time l at the forecast generation time t.

5.5.4 Experimental Settings

For each method, its hyperparameters are tuned using the validation set. The hyperparameter for ridge regression corresponds to the ridge regularizer. For DNN-type approaches including FFN, LSTM, GFFN, GLSTM, DARNN, MTL-LSTM and the proposed MSH-LSTM, the number of nodes in each hidden layer is a hyperparameter that needs to be tuned. For all the methods, the

number of nodes is varied from 10 to 50. Other hyperparameters include batch size, initial learning rate and dropout rate are also tuned independently for each dataset based on its performance on the corresponding validation set. Furthermore, as the loss function for DNN is non-convex, different initialization of the model parameters may yield different solutions. Consequently, we test each hyperparameter setting for the FFN, LSTM, GFFN, GLSTM, MTL-LSTM and MSH-LSTM with 15 different initializations of the weights. RMSE values are reported based on their average over these 15 runs.

5.5.5 Results and Discussion

A summary of the RMSE values, averaged across the 6 data splits, is shown in Table 5.3. In terms of their overall RMSE, simple baseline methods such as ensemble mean and ARIMA have the worst performance among all the methods. This shows the importance of combining the ensemble member forecasts in a weighted fashion to obtain better predictions instead of using only the mean forecasts or historical time series alone for making long-term predictions. The next worst performer is ridge regression, which suggests the importance of using non-linear approaches to aggregate the ensemble predictions. Furthermore, global models such as GFFN and GLSTM also have worse RMSE compared to their independent local model counterparts (FFN and LSTM). This suggests there is significant difference in the skills of the individual physical models for making predictions at different lead times, which explains the inferior performance of the one-size-fits-all global models.

Among all the competing methods, MSH-LSTM achieves the lowest overall RMSE, which demonstrates the effectiveness of the proposed framework for the multi-step-ahead ensemble forecasting problem. In particular, it outperforms both MTL-LSTM, which is based on a conventional approach to multi-task deep learning [105] and DARNN, which is the state-of-art DNN-typed hierarchical approach for time series prediction with exogenous variables [100]. The proposed MSH-LSTM framework also outperforms both MTL-LSTM and DARNN for the majority of the lead times except for lead times 0 and 8. The effectiveness of MSH-LSTM can be explained as it is the only framework that considers lead-time level autocorrelation, whereas other frameworks

such as MTL-LSTM and DARNN only account for generation-time level autocorrelation and the relationship between predictors. The result also suggests that the hierarchical LSTM architecture of MSH-LSTM is more suitable for the problem than the hard parameter sharing strategy employed by MTL-LSTM.

In terms of lead-time RMSE, MSH-LSTM has the lowest RMSE for lead times 1 to 5 and has among the top-3 lowest RMSE for lead times 6, 7, and 8. Both MSH-LSTM and MTL-LSTM appear to perform slightly worse than conventional LSTM for lead time 0. One possible explanation is that, since both MSH-LSTM and MTL-LSTM are multi-task learning approaches, the long-term forecasts performance may be improved at the expense of a slight degradation in their accuracy for forecasting lead time 0. In addition, MSH-LSTM is outperformed by FFN for lead times 6 to 8, though by relatively small margin.

Another interesting observation is that the independent LSTM models generally do a much better job at short-term forecasting but perform worse at long-term forecasting (4 months or more) compared to independent FFN models. In both approaches, the models are trained independently for each lead time task. As the longer-term predictors have higher variance in the ensemble forecasts, this suggests that LSTM may not be as effective dealing with higher variance in the predictors compared to FFN. This limitation of LSTM also seems to affect the performance of MTL-LSTM and MSH-LSTM. Nevertheless, it appears that MSH-LSTM is able to compensate for such limitation by regularizing its predictions to ensure the generation-time and lead-time level autocorrelations are modeled. Overall, its RMSE performance is comparable to FFN for longer lead time forecasts (4 months or more).

The previous analysis compares the performance of different methods in terms of their overall and lead-time specific RMSE. The reported RMSE values are averaged over all 58 grid cells in the data. To determine how well each method performs on the grid cells, Table 5.4 summarizes the percentage of grid cells in which the method specified in the given row has lower RMSE than the method specified by the column. The results show that MSH-LSTM outperforms all the baseline methods in more than 70% of the grid cells. Furthermore, although the RMSE difference shown

Table 5.3: Comparison of RMSE values among the competing methods for all 9 forecast lead times

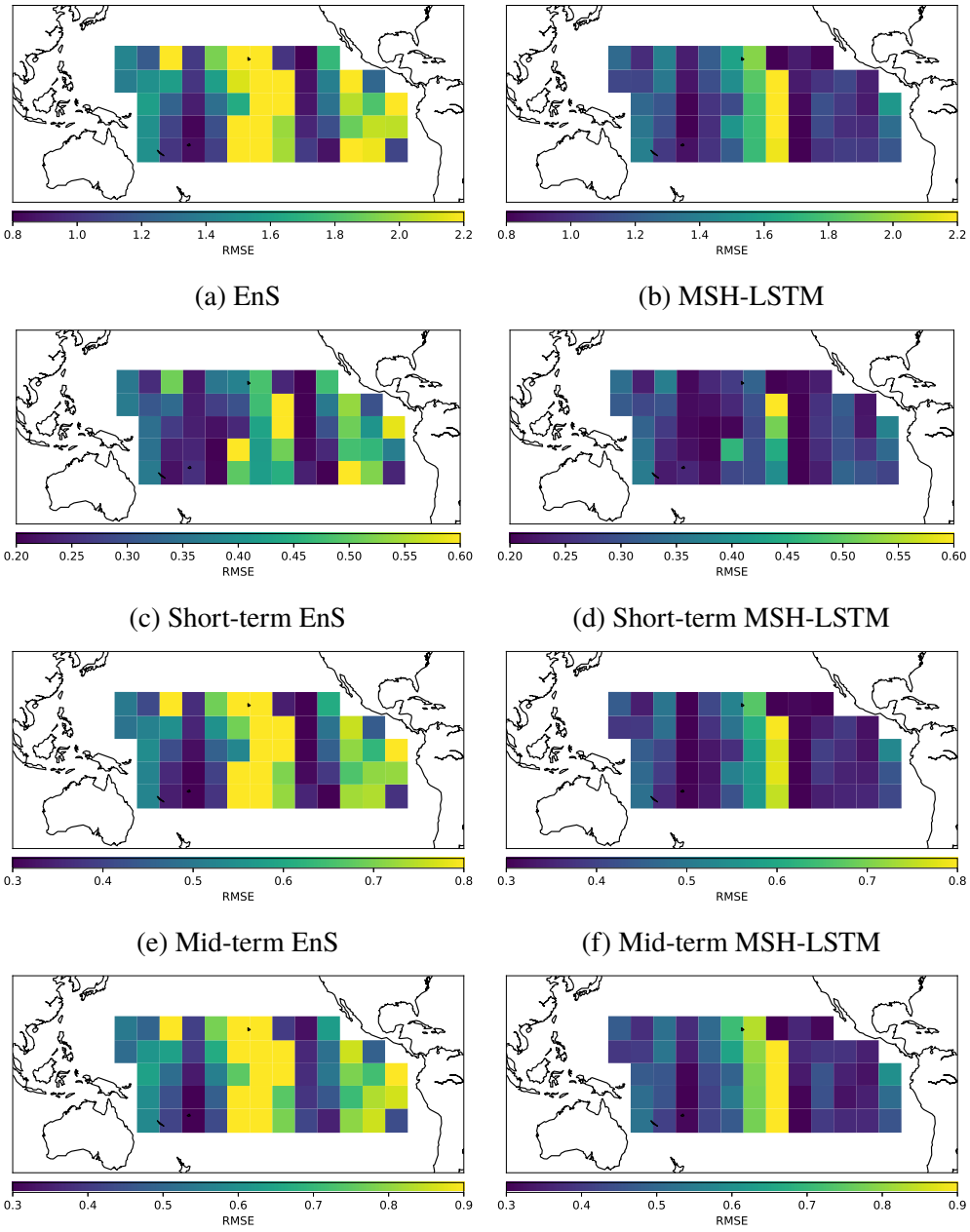
Lead	EnS	Ridge	GFFN	GLSTM	FFN	LSTM	MTL-LSTM	ARIMA	DARNN	MSH-LSTM
0	0.2652	0.2811	0.2830	0.2787	0.2129	**0.1996**	0.2090	0.2592	0.2610	0.2173
1	0.4322	0.3940	0.3652	0.3560	0.3435	0.3230	0.3313	0.3706	0.3475	**0.3057**
2	0.5307	0.4437	0.4058	0.3974	0.4015	0.3827	0.3890	0.4414	0.4053	**0.3621**
3	0.5933	0.4697	0.4313	0.4286	0.4286	0.4176	0.4302	0.4901	0.4399	**0.4032**
4	0.6402	0.4900	0.4574	0.4633	0.4548	0.4648	0.4735	0.5279	0.4747	**0.4426**
5	0.6757	0.5058	0.4794	0.4965	0.4775	0.5029	0.5050	0.5529	0.5132	**0.4744**
6	0.7039	0.5205	0.4969	0.5185	**0.4916**	0.5264	0.5260	0.5701	0.5155	0.5019
7	0.7268	0.5302	0.5151	0.5344	**0.5097**	0.5367	0.5326	0.5827	0.5333	0.5192
8	0.7458	0.5407	0.5400	0.5546	0.5298	0.5519	0.5445	0.5923	**0.5245**	0.5355
overall	1.8268	1.4114	1.3443	1.3673	1.3136	1.3432	1.3514	1.4964	1.3641	**1.2898**

Table 5.4: A win-loss table comparing the performance of the competing methods across all 58 grid cells. Each (i, j)-th entry in the table represents the fraction of grid cells in which method i has lower RMSE than method j.

	EnS	Ridge	GFFN	GLSTM	FFN	LSTM	MTL-LSTM	ARIMA	DARNN	MSH-LSTM
EnS	0	0.2586	0.1379	0.1724	0.1552	0.1897	0.1897	0.3621	0.2414	0.1552
Ridge	0.7414	0	0.1724	0.2241	0.1897	0.2586	0.2069	0.6207	0.2931	0.1034
GFFN	0.8621	0.8276	0	0.5862	0.4310	0.5345	0.5517	0.8793	0.6379	0.2586
GLSTM	0.8276	0.7759	0.4138	0	0.4138	0.4138	0.5172	0.8621	0.5517	0.2241
FFN	0.8448	0.8103	0.5690	0.5862	0	0.6207	0.6379	0.8448	0.7241	0.2414
LSTM	0.8103	0.7414	0.4655	0.5862	0.3793	0	0.5172	0.8793	0.6207	0.2931
MTL-LSTM	0.8103	0.7931	0.4483	0.4828	0.3621	0.4828	0	0	0.6724	0.2241
ARIMA	0.6379	0.3793	0.1207	0.1379	0.1552	0.1207	0.1379	0	0.1724	0.0345
DARNN	0.7586	0.7069	0.3621	0.4483	0.2759	0.3793	0.3276	0.8276	0	0.0690
MSH-LSTM	0.8448	0.8966	0.7414	0.7759	0.7586	0.7069	0.7759	0.9655	0.9310	0

in Table 5.3 for MSH-LSTM and FFN is not that large, MSH-LSTM actually outperforms FFN for more than 75% of the grid cells. MSH-LSTM also outperformed MTL-LSTM (by more than 77%) and DARNN (by more than 93%), which demonstrates the effectiveness of proposed MSH-LSTM framework for the multi-step-ahead ensemble SST forecasting problem.

To illustrate the performance improvement achieved by MSH-LSTM in different grid cells, Fig. 5.3 shows a map of the RMSE values for ensemble mean and MSH-LSTM, where lighter (yellow) color indicates higher RMSE values and darker (blue) color indicates lower RMSE. The maps were plotted for different lead times. Figs. 5.3(a) and 5.3(b) correspond to the the overall RMSE for all 9 lead times. Figs. 5.3(c) and 5.3(d) correspond to short-term forecasts (from 0 to 2 months), 5.3(e) and 5.3(f) are for mid-term forecasts (between 3 to 5 months lead time), and



(a) EnS (b) MSH-LSTM

(c) Short-term EnS (d) Short-term MSH-LSTM

(e) Mid-term EnS (f) Mid-term MSH-LSTM

(g) Long-term EnS (h) Long-term MSH-LSTM

Figure 5.3: Performance on each grid cell by EnS and MSH-LSTM.

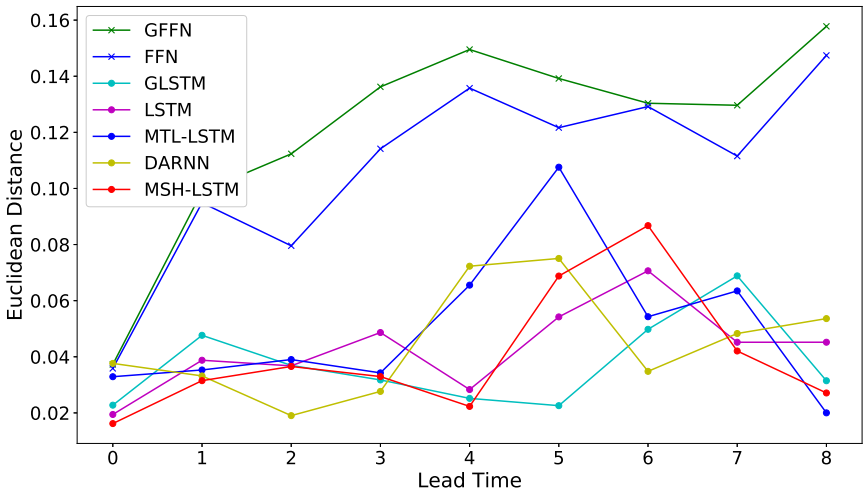
5.3(g) and 5.3(h) for long-term forecasts (more than 6 months). The maps show that MSH-LSTM outperforms the ensemble median in the majority of the grid cells, especially for mid-term and long-term predictions.

Fig 5.4 shows the amount that the temporal autocorrelation of the ground truth time series is

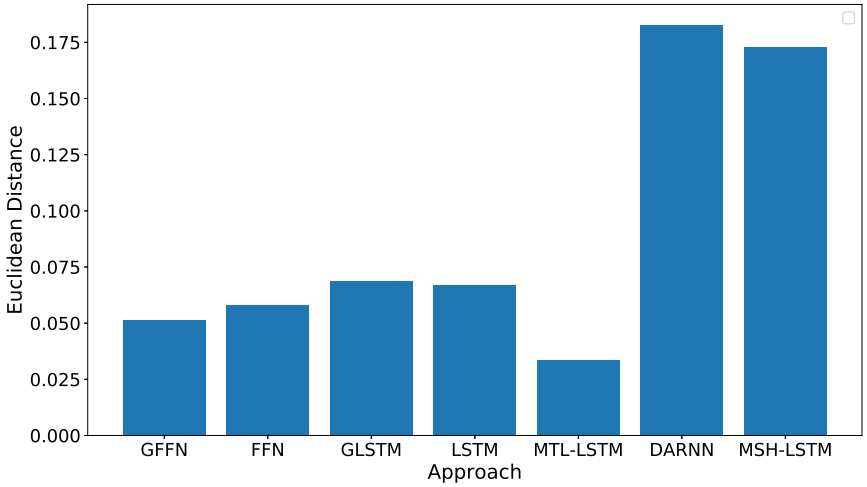
preserved by each approach. The temporal autocorrelation for any given time series at a lag k, $ACF(k)$, is obtained by shifting its sequence of values by k time steps (lags) and computing the correlation between the shifted sequence and the unshifted one. If there are p such shifts, this produces an autocorrelation vector of length p, i.e., $[ACF(1), ACF(2), \cdots, ACF(p)]$, which is then compared against the autocorrelation vector of the ground truth time series. The degree to which temporal autocorrelation is preserved can be determined by taking the Euclidean distance between the two vectors.

Fig. 5.1 depicts two types of temporal autocorrelations that must be preserved by multi-step-ahead time series forecasting methods—lead-time level and generation-time level autocorrelation. For generation-time level autocorrelation, the results shown in Fig. 5.4a suggest that LSTM and its variants, including the proposed MSH-LSTM approach, closely model the temporal autocorrelation structure of the ground truth SST time series as the Euclidean distances calculated for LSTM-based approaches are relatively smaller compared to non-LSTM methods such as FFN and GFNN. This is not surprising as the LSTM-based methods are designed to capture the temporal dependencies of the forecasts generated for different time steps, while the non-LSTM methods are not designed for modeling temporal dependencies of the data.

For lead-time level autocorrelation, at first glance, the results shown in Fig. 5.4b appear to suggest that both MSH-LSTM and DARNN do not capture the lead-time level autocorrelation as effectively as other baseline methods. To further illustrate this, Fig. 5.5 shows the correlogram plots for each forecasting method as well as for the ground truth SST time series. Notice that the lead-time level autocorrelation for MSH-LSTM and DARNN are much higher than other approaches and the ground truth, as the number of lags increases. Despite over-estimating the lead-time level temporal autocorrelation, the RMSE results shown in Table 5.3 suggest that MSH-LSTM was able to exploit the higher autocorrelation at longer lags and in this way improve its long-term forecasts. Similarly, the results in Table 5.3 also show that DARNN reaches the best RMSE at lead time 8, which is consistent with it having the highest temporal autocorrelation at lag 8. However, the RMSE of DARNN is worse than MSH-LSTM and other baselines at other lead times even though its



(a) Generation time level temporal autocorrelation.



(b) Lead time level temporal autocorrelation

Figure 5.4: Comparison between the temporal autocorrelation of the proposed MSH-LSTM framework and other baseline methods.

autocorrelation is still high. This suggests that MSH-LSTM was able to leverage its high lead-time autocorrelation to improve its prediction accuracy more effectively than DARNN.

Finally, we evaluate the importance of each physical model for different lead time tasks. The models of both ridge regression and MSH-LSTM are analyzed. For ridge regression, the importance of each model can be evaluated by examining the magnitude of coefficients. For MSH-LSTM, the gradient of the loss w.r.t each physical model forecast is computed [111], and the distribution of the gradient magnitudes is investigated. In Figure 5.6, the gradients for the 7 physical models are

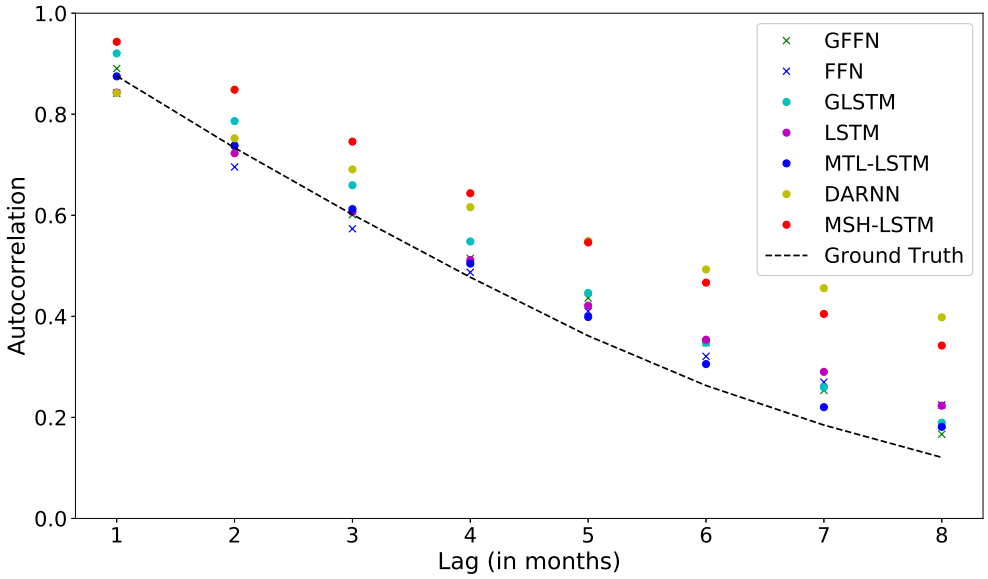
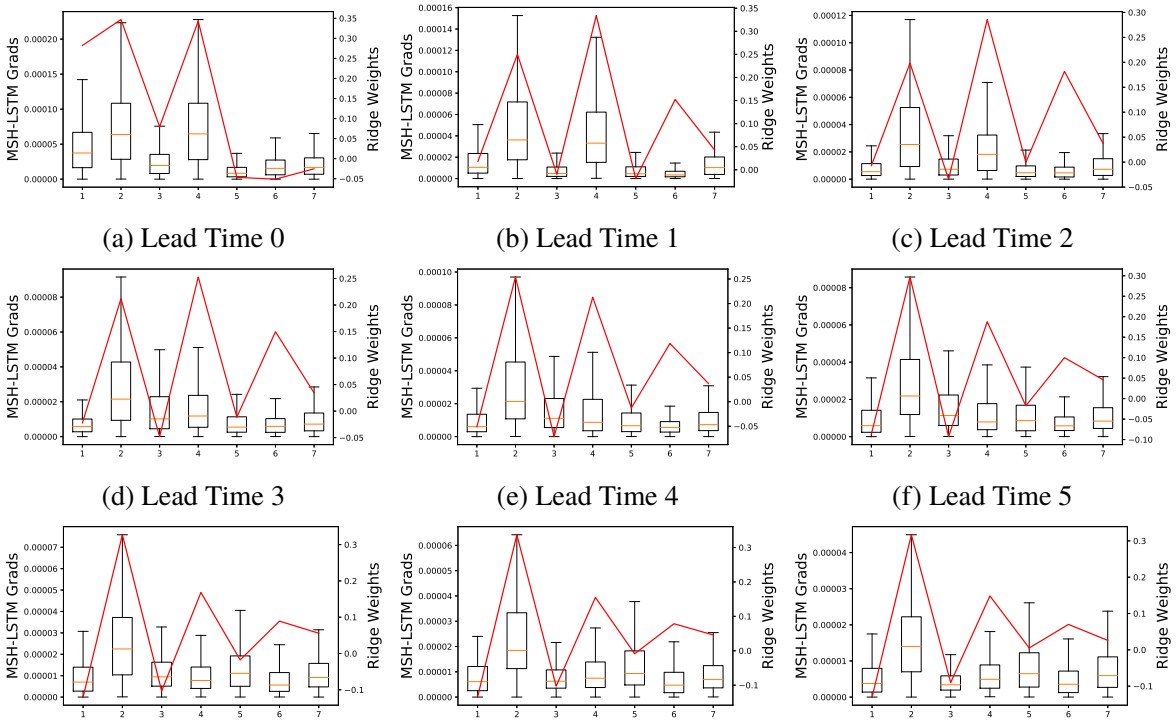


Figure 5.5: Correlogram plots for lead-time level autocorrelation of MSH-LSTM and other methods (including the ground truth SST time series).

illustrated by box plots while the magnitude of the ridge regression coefficients is illustrated with the red curve. For both ridge regression and the MSH-LSTM the most important physical model tends to be model 2 (CMC2-CanCM4). Model 4 also has a relatively large importance for both MSH-LSTM on shorter-term forecasting models.

5.6 Summary

In this chapter, a novel multi-task neural network architecture is proposed to address the multi-step-ahead time series forecasting problem. The proposed framework considers each lead time forecast as a separate learning task and employs a hierarchical LSTM structure to capture both lead-time and generation-time level autocorrelation of the data. The effectiveness of the proposed architecture is evaluated on a 29-year monthly sea surface temperature data from the North American Multi-Model Ensemble (NMME) project. The results showed that the proposed method outperformed existing hierarchical and non-hierarchical neural network, MTL, and other conventional time series prediction methods.



(a) Lead Time 0 (b) Lead Time 1 (c) Lead Time 2

(d) Lead Time 3 (e) Lead Time 4 (f) Lead Time 5

(g) Lead Time 6 (h) Lead Time 7 (i) Lead Time 8

Figure 5.6: Gradient distribution of each model for different lead time tasks. The x-axis represents the indices of physical models that are listed in Table. 5.1. The box plots are gradients distribution of each model for MSH-LSTM. The red curve are the computed ridge regression coefficients.

CHAPTER 6

CONCLUSIONS & FUTURE WORK

6.1 Summary of Thesis Contributions

With the growing complexity and prevalence of spatial and temporal data from various disciplines, it has become an important but more challenging research topic to learn accurate predictive models from such data. For some complex prediction tasks, learning a single model to all observations is often undesirable as such a model may not capture the intricate details and variabilities across different samples.

In this thesis, I present several innovative frameworks of multi-task learning techniques on some real-world applications on spatial and temporal data, including activity recognition in multi-modal sensor data to large-scale climate and sea surface temperature predictions. I investigate into their related problems, examine the fundamental challenges, and propose novel MTL frameworks for solution. The contributions of this thesis are summarized as below.

In Chapter 3, a probabilistic MTL framework was designed for multi-modal time series classification problem at multiple locations. The framework was motivated by the need to address both temporal and spatial dependencies of the classes as well as the block missing value problem, which arises due to the varying types of features (modalities) available at different locations. As proof of concept, the proposed framework was applied to the activity recognition problem, where the task is to identify user activities (e.g., walking, sitting, jumping, or lying down) in multiple rooms using multi-modal sensor data from accelerometer, video cameras, and environmental sensors. Experimental results showed that the proposed framework outperformed baseline methods including K-nearest neighbor, Conditional Random Field, and single-task learning with multinomial softmax regression. In addition to improving classification performance, the framework also produces a data-driven transition matrix between the activities as byproduct.

In Chapter 4, a novel MTL framework was proposed to address the challenges of time series

forecasting at multiple locations simultaneously. Unlike conventional MTL approaches, the proposed framework was designed to preserve the marginal distribution of the response variable in addition to point-wise accuracy. Specifically, it employs a Parzen window based kernel density estimation (KDE) approach to compute the probability density function and an L^2-distance measure to determine the discrepancy between the true and predicted distributions. When using on a real-world climate dataset as case study, experimental results showed that the proposed framework outperformed existing non-distribution preserving methods in more than 78% of the weather stations considered in this study. It also outperformed another baseline distribution preserving method (with single-task learning) in more than 78% of the weather stations.

Finally, a multi-task hierarchical LSTM architecture was designed in Chapter 5 to deal with the challenges proposed in multi-step-ahead time series prediction. The proposed architecture was designed to improve forecasting of sea surface temperature at longer lead times. The proposed architecture considers each lead time as a separate learning task and employs a two-level hierarchical LSTM structure to capture both the lead-time and forecast generation time dependencies of the data. Experimental results using 29 years of monthly sea surface temperature data from the North American Multi-Model Ensemble (NMME) project demonstrated the efficacy of the proposed method compared to 9 other baseline methods, including ridge regression, MTL, and other conventional neural network approaches.

6.2 Future Research Directions

This dissertation has demonstrated the efficacy of using MTL to improve performance of predictive modeling in spatial and temporal data. The success of MTL in the application domains investigated in this study lay forth to several potential future research directions to pursue.

First, the proposed multi-modal time series classification approach is limited to a linear model using softmax regression. Extending the approach to nonlinear models via deep neural networks will be an interesting research direction. However, there are several key questions that must be answered in order to facilitate the development of such an approach. Among them include: "How to

incorporate the spatial and temporal dependencies of the classes?", "How to deal with the varying feature types available for different locations?", and "How to relate the modeling for different rooms without increasing the number of model parameters significantly?" One possibility is to extend the hierarchical LSTM approach described in Chapter 5 to incorporate the classifier properties of the probabilistic MTL framework introduced in Chapter 3. From the application perspective, the current activity recognition framework assumes there is only a single subject to be monitored. Extending such an approach to multiple subject setting is another potentially interesting research direction. Demographical information can be used to build relationships between subjects, and it will provide more customized model for each subject.

Second, for the multi-location time series forecasting problem, the distribution-preserving multi-task regression framework proposed in Chapter 4 also considers only a linear model. Extending the distribution preserving approach to deep networks is another potentially interesting research direction. The non-identical distributed property exists in both spatial and temporal dimensions. In Chapter 4, the distribution preserving mechanism only keeps the temporal distribution contour at each location, without accounting for the spatial distribution at each fixed time. For example, the spatial distribution of precipitation certainly varies from spring to winter. An extension of this work by taking account of the distribution property on spatial dimension will improve the interpretability and performance of predictions. The approach can potentially be enhanced to a 3D-distribution setting, e.g., by defining a Parzen window and L^2-distance over both space and time, to ensure the spatial distribution of the response variable is also preserved by the prediction model.

Finally, this thesis mainly discusses the predictive modeling under an offline setting. Nevertheless, in practice, data keeps growing rapidly on both spatial and temporal dimension. When observations are collected from a new location or a new time step, re-training the entire model in a batch way would be very expensive. What is more, cold start problem should also be considered for previously unseen locations. As a result, the framework should consider the data increment without requiring re-training from scratch. Incremental learning mechanism should be potentially incorporated to the existing model for a runtime execution.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Google ads preferences manager. http://google.com/ads/preferences.

[2] Instagram. http://www.instagram.com.

[3] The sphere challenge: Activity recognition with multimodal sensor data. http://blog.drivendata.org/2016/06/06/sphere-benchmark/.

[4] Transfer learning. https://www.youtube.com/watch?v=qD6iD4TFsdQ.

[5] Zubin Abraham, Pang-Ning Tan, Perdinan, Julie Winkler, Shiyuan Zhong, and Malgorzata Liszewska. Distribution regularized regression framework for climate modeling. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 333–341. SIAM, 2013.

[6] Zubin Abraham, Pang-Ning Tan, Perdinan, Julie Winkler, Shiyuan Zhong, and Malgorzata Liszewska. Position preserving multi-output prediction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 320–335. Springer, 2013.

[7] Alekh Agarwal, Alexander Rakhlin, and Peter Bartlett. Matrix regularization techniques for online multitask learning. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-138*, 2008.

[8] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in neural information processing systems*, pages 41–48, 2007.

[9] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[10] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. Spatio-temporal data mining: A survey of problems and methods. *arXiv preprint arXiv:1711.04710*, 2017.

[11] Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. *arXiv preprint arXiv:1802.09913*, 2018.

[12] Jonghun Baek, Geehyuk Lee, Wonbae Park, and Byoung-Ju Yun. Accelerometer signal processing for user activity detection. In *Proceedings of International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 610–617, 2004.

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

[14] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *Proceedings of International Conference on Pervasive Computing*, pages 1–17. Springer, 2004.

[15] Steffen Bickel, Jasmina Bogojeska, Thomas Lengauer, and Tobias Scheffer. Multi-task learning for hiv therapy screening. In *Proceedings of the 25th international conference on Machine learning*, pages 56–63. ACM, 2008.

[16] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[17] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[18] Paul Bromiley. Products and convolutions of gaussian probability density functions. *Tina-Vision Memo*, 3(4):1, 2003.

[19] J Solai Carós, O Chételat, Patrick Celka, S Dasen, and J CmÃral. Very low complexity algorithm for ambulatory activity classification. In *Proceedings of the 3rd European Medical and Biological Conference EMBEC*, pages 16–20, 2005.

[20] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[21] R Caruna. Multitask learning: A knowledge-based source of inductive bias. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 41–48, 1993.

[22] Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11(Oct):2901–2934, 2010.

[23] Junghoon Chae, Dennis Thom, Harald Bosch, Yun Jang, Ross Maciejewski, David S Ebert, and Thomas Ertl. Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 143–152. IEEE, 2012.

[24] Jianhui Chen, Ji Liu, and Jieping Ye. Learning incoherent sparse and low-rank patterns from multiple tasks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):22, 2012.

[25] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 42–50. ACM, 2011.

[26] Kevin KW Cheung. A review of ensemble forecasting techniques with a focus on tropical cyclone forecasting. *Meteorological Applications*, 8(3):315–332, 2001.

[27] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[28] Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems*, pages 1647–1655, 2011.

[29] TM Darcey and PD Williamson. Spatio-temporal eeg measures and their application to human intracranially recorded epileptic seizures. *Electroencephalography and clinical neurophysiology*, 61(6):573–587, 1985.

[30] Richard A Dare and John L McBride. Sea surface temperature response to tropical cyclones. *Monthly Weather Review*, 139(12):3798–3808, 2011.

[31] Jeffery Jonathan Davis, Robert Kozma, Chin-Teng Lin, and Walter J Freeman. Spatio-temporal eeg pattern extraction using high-density scalp arrays. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 889–896. IEEE, 2016.

[32] Krzysztof Dembczynski, Arkadiusz Jachnik, Wojciech Kotlowski, Willem Waegeman, and Eyke Hüllermeier. Optimizing the f-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *International Conference on Machine Learning*, pages 1130–1138, 2013.

[33] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012.

[34] Xavier Descombes, Frithjof Kruggel, and D Yves Von Cramon. Spatio-temporal fmri analysis using markov random fields. *IEEE transactions on medical imaging*, 17(6):1028–1039, 1998.

[35] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[36] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference of Machine Learning*, volume 12, pages 194–202, 1995.

[37] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 845–850, 2015.

[38] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[39] Anders Eklund, Thomas E Nichols, and Hans Knutsson. Cluster failure: why fmri inferences for spatial extent have inflated false-positive rates. *Proceedings of the National Academy of Sciences*, page 201602413, 2016.

[40] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[41] Yuchun Fang, Zhengyan Ma, Zhaoxiang Zhang, Xu-Yao Zhang, and Xiang Bai. Dynamic multi-task learning with convolutional neural network. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, pages 19–25, 2017.

[42] Elizabeth A Franz, James C Eliassen, Richard B Ivry, and Michael S Gazzaniga. Dissociation of spatial and temporal coupling in the bimanual movements of callosotomy patients. *Psychological Science*, 7(5):306–310, 1996.

[43] R Friedrich, A Fuchs, and H Haken. Spatio-temporal eeg patterns. In *Rhythms in physiological systems*, pages 315–338. Springer, 1991.

[44] Ankit Gandhi, Arjun Sharma, Arijit Biswas, and Om Deshmukh. Gethr-net: A generalized temporally hybrid recurrent neural network for multimodal information fusion. In *European Conference on Computer Vision*, pages 883–899. Springer, 2016.

[45] Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Chai. Physical causality of action verbs in grounded language understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1814–1824, 2016.

[46] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[47] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[48] Yuanhong Guan, Jieshun Zhu, Bohua Huang, Zeng-Zhen Hu, and James L Kinter III. South pacific ocean dipole: A predictable mode on multiseasonal time scales. *Journal of Climate*, 27(4):1648–1658, 2014.

[49] D Herring. What is el nino? *NASA's Earth Observatory*, 2009.

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

[51] Bo Hu, Mohsen Jamali, and Martin Ester. Spatio-temporal topic modeling in mobile social media for location recommendation. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1073–1078. IEEE, 2013.

[52] Zeng-Zhen Hu, Arun Kumar, Bohua Huang, Wanqiu Wang, Jieshun Zhu, and Caihong Wen. Prediction skill of monthly sst in the north atlantic ocean in ncep climate forecast system version 2. *Climate dynamics*, 40(11-12):2745–2759, 2013.

[53] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7304–7308. IEEE, 2013.

[54] Yan Huang, Wei Wang, Liang Wang, and Tieniu Tan. Multi-task deep neural network for multi-label learning. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 2897–2900. IEEE, 2013.

[55] Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th annual international conference on machine learning*, pages 457–464. ACM, 2009.

[56] Eugenia Kalnay, Masao Kanamitsu, Robert Kistler, William Collins, Dennis Deaven, Lev Gandin, Mark Iredell, Suranjana Saha, Glenn White, John Woollen, et al. The ncep/ncar 40-year reanalysis project. *Bulletin of the American meteorological Society*, 77(3):437–471, 1996.

[57] Anuj Karpatne, James Faghmous, Jaya Kawale, Luke Styles, Mace Blank, Varun Mithal, Xi Chen, Ankush Khandelwal, Shyam Boriah, Karsten Steinhaeuser, et al. Earth science applications of sensor data. In *Managing and Mining Sensor Data*, pages 505–530. Springer, 2013.

[58] Harry H Kelejian and Ingmar R Prucha. A generalized spatial two-stage least squares procedure for estimating a spatial autoregressive model with autoregressive disturbances. *The Journal of Real Estate Finance and Economics*, 17(1):99–121, 1998.

[59] Eunju Kim, Sumi Helal, and Diane Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1), 2010.

[60] Minyoung Kim. Semi-supervised learning of hidden conditional random fields for time-series classification. *Neurocomputing*, 119:339–349, 2013.

[61] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[62] Ben P Kirtman, Dughong Min, Johnna M Infanti, James L Kinter III, Daniel A Paolino, Qin Zhang, Huug Van Den Dool, Suranjana Saha, Malaquias Pena Mendez, Emily Becker, et al. The north american multimodel ensemble: phase-1 seasonal-to-interannual prediction; phase-2 toward developing intraseasonal prediction. *Bulletin of the American Meteorological Society*, 95(4):585–601, 2014.

[63] Slava Kisilevich, Florian Mansmann, Mirco Nanni, and Salvatore Rinzivillo. Spatio-temporal clustering. In *Data mining and knowledge discovery handbook*, pages 855–874. Springer, 2009.

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

[65] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[66] Pierre Legendre. Spatial autocorrelation: trouble or new paradigm? *Ecology*, 74(6):1659–1673, 1993.

[67] Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello. A practical approach to recognizing physical activities. In *Proceedings of the International Conference on Pervasive Computing*, pages 1–16. Springer, 2006.

[68] Caiyan Li and Hongzhe Li. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24(9):1175–1182, 2008.

[69] Sijin Li, Zhi-Qiang Liu, and Antoni B Chan. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 482–489, 2014.

[70] Lei Liu, Prakash Mandayam Comar, Sabyasachi Saha, Pang-Ning Tan, and Antonio Nucci. Recursive nmf: Efficient label tree learning for large multi-class problems. In *IEEE International Conf on Pattern Recognition*, 2012.

[71] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1010–1018. ACM, 2011.

[72] Xi Liu, Lei Liu, Steven J Simske, and Jerry Liu. Human daily activity recognition for healthcare using wearable and visual sensing data. In *Proceedings of IEEE International Conference on Healthcare Informatics (ICHI-2016)*, pages 24–31. IEEE, 2016.

[73] Xi Liu, Lei Liu, and Pang-Ning Tan. Location-based hierarchical approach for activity recognition with multi-modal sensor data. In *ECML/PKDD 2016 Discovery Challenge, Riva del Gada, Italy (2016)*, pages 24–31. IEEE, 2016.

[74] Xi Liu, Han Hee Song, Mario Baldi, and Pang-Ning Tan. Macro-scale mobile app market analysis using customized hierarchical categorization. In *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2016.

[75] Xi Liu, Pang-Ning Tan, Zubin Abraham, Lifeng Luo, and Pouyan Hatami. Distribution preserving multi-task regression for spatio-temporal data. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1134–1139. IEEE, 2018.

[76] Xi Liu, Pang-Ning Tan, and Lei Liu. Stars: Soft multi-task learning for activity recognition from multi-modal sensor data. In *Proceedings of Pacific Asian Conference on Knowledge Discovery and Data Mining (PAKDD-2018)*, pages 571–583. Springer, 2018.

[77] Ye Liu, Yu Zheng, Yuxuan Liang, Shuming Liu, and David S Rosenblum. Urban water quality prediction based on multi-task multi-view learning. 2016.

[78] Jing-Jia Luo, Sebastien Masson, Swadhin Behera, and Toshio Yamagata. Experimental forecasts of the indian ocean dipole using a coupled oagcm. *Journal of climate*, 20(10):2178–2190, 2007.

[79] Yong Luo, Dacheng Tao, Bo Geng, Chao Xu, and Stephen J Maybank. Manifold regularized multitask learning for semi-supervised multilabel image classification. *IEEE Transactions on Image Processing*, 22(2):523–536, 2013.

[80] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[81] Spyros Makridakis and Michele Hibon. Arma models and the box–jenkins methodology. *Journal of Forecasting*, 16(3):147–163, 1997.

[82] Douglas Maraun. Bias correction, quantile mapping, and downscaling: Revisiting the inflation issue. *Journal of Climate*, 26(6):2137–2143, 2013.

[83] Yasuko Matsubara, Yasushi Sakurai, Willem G Van Panhuis, and Christos Faloutsos. Funnel: automatic mining of spatially coevolving epidemics. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 105–114. ACM, 2014.

[84] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

[85] MJ Menne, I Durre, B Korzeniewski, S McNeal, K Thomas, X Yin, S Anthony, R Ray, RS Vose, BE Gleason, et al. Global historical climatology network-daily (ghcn-daily), version 3.22. noaa national climatic data center, 2016.

[86] Microsoft Kinect SDK. `https://developer.microsoft.com/en-us/windows/kinect`.

[87] Harvey J Miller. Tobler's first law and spatial analysis. *Annals of the Association of American Geographers*, 94(2):284–289, 2004.

[88] Wanli Min and Laura Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606–616, 2011.

[89] Phan Q Minh, Roger S Morris, Birgit Schauer, Mark Stevenson, Jackie Benschop, Hoang V Nam, and Ron Jackson. Spatio-temporal epidemiology of highly pathogenic avian influenza outbreaks in the two deltas of vietnam during 2003–2007. *Preventive veterinary medicine*, 89(1-2):16–24, 2009.

[90] Pradeep Mohan, Shashi Shekhar, James A Shine, and James P Rogers. Cascading spatio-temporal pattern discovery: A summary of results. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 327–338. SIAM, 2010.

[91] Margaret A Oliver and Richard Webster. Kriging: a method of interpolation for geographical information systems. *International Journal of Geographical Information System*, 4(3):313–332, 1990.

[92] Open Source Computer Vision Library. `http://opencv.org/`.

[93] OpenNI. `http://www.openni.org/`.

[94] Jiahao Pang and Gene Cheung. Graph laplacian regularization for image denoising: Analysis in the continuous domain. *IEEE Transactions on Image Processing*, 26(4):1770–1785, 2017.

[95] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch, 2017.

[96] Jian Peng, Sha Chen, Huiling Lü, Yanxu Liu, and Jiansheng Wu. Spatiotemporal patterns of remotely sensed pm2. 5 concentration in china from 1999 to 2011. *Remote Sensing of Environment*, 174:109–121, 2016.

[97] Steven M Pincus, Igor M Gladstone, and Richard A Ehrenkranz. A regularity statistic for medical data analysis. *Journal of clinical monitoring*, 7(4):335–345, 1991.

[98] Point Cloud Library. `http://www.pointclouds.org`.

[99] R Poulin and S Morand. Geographical distances and the similarity among parasite communities of conspecific host populations. *Parasitology*, 119(4):369–374, 1999.

[100] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.

[101] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

[102] K Venkateswara Rao, A Govardhan, and KV Chalapati Rao. Spatiotemporal data mining: Issues, tasks and applications. *International Journal of Computer Science and Engineering Survey*, 3(1):39, 2012.

[103] Thomas Reichler and Junsu Kim. How well do coupled models simulate today's climate? *Bulletin of the American Meteorological Society*, 89(3):303–312, 2008.

[104] Jose Antonio MR Rocha, Valéria C Times, Gabriel Oliveira, Luis O Alvares, and Vania Bogorny. Db-smot: A direction-based spatio-temporal clustering method. In *2010 5th IEEE international conference intelligent systems*, pages 114–119. IEEE, 2010.

[105] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[106] Patrick H Ryan, Grace K LeMasters, Pratim Biswas, Linda Levin, Shaohua Hu, Mark Lindsey, David I Bernstein, James Lockey, Manuel Villareal, Gurjit K Khurana Hershey, et al. A comparison of proximity and land use regression traffic exposure models and wheezing in infants. *Environmental health perspectives*, 115(2):278, 2007.

[107] Shashi Shekhar, Zhe Jiang, Reem Ali, Emre Eftelioglu, Xun Tang, Venkata Gunturi, and Xun Zhou. Spatiotemporal data mining: a computational perspective. *ISPRS International Journal of Geo-Information*, 4(4):2306–2338, 2015.

[108] Shashi Shekhar, Chang-Tien Lu, and Pusheng Zhang. A unified approach to detecting spatial outliers. *GeoInformatica*, 7(2):139–166, 2003.

[109] Amir Shmuel, Essa Yacoub, Denis Chaimow, Nikos K Logothetis, and Kamil Ugurbil. Spatio-temporal point-spread function of fmri signal in human gray matter at 7 tesla. *Neuroimage*, 35(2):539–552, 2007.

[110] Martin B Short, Maria R D'orsogna, Virginia B Pasour, George E Tita, Paul J Brantingham, Andrea L Bertozzi, and Lincoln B Chayes. A statistical model of criminal behavior. *Mathematical Models and Methods in Applied Sciences*, 18(supp01):1249–1267, 2008.

[111] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

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

[113] John R Smith, Jelena Tesic, and Rong Yan. Method and apparatus for model-shared subspace boosting for multi-label classification, June 7 2011. US Patent 7,958,068.

[114] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 231–235, 2016.

[115] Masashi Sugiyama, Song Liu, Marthinus Christoffel Du Plessis, Masao Yamanaka, Makoto Yamada, Taiji Suzuki, and Takafumi Kanamori. Direct divergence approximation between probability distributions and its applications in machine learning. *Journal of Computing Science and Engineering*, 7(2):99–111, 2013.

[116] Pei Sun and Sanjay Chawla. On local spatial outliers. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 209–216. IEEE, 2004.

[117] Xu Sun, Hisashi Kashima, and Naonori Ueda. Large-scale personalized human activity recognition using online multitask learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2551–2563, 2013.

[118] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. *Introduction to Statistical Relational Learning*, pages 93–128, 2006.

[119] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Addison Wesley, 2006.

[120] Bridget Thrasher, Edwin P. Maurer, C. McKellar, and P. B. Duffy. Bias correcting climate model simulated daily temperature extremes with quantile mapping. *Hydrology and Earth System Sciences*, 16(9):3309, 2012.

[121] Waldo R Tobler. A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240, 1970.

[122] Niall Twomey, Tom Diethe, Meelis Kull, Hao Song, Massimo Camplani, Sion Hannuna, Xenofon Fafoutis, Ni Zhu, Pete Woznowski, Peter Flach, and Ian Craddock. The SPHERE challenge: Activity recognition with multimodal sensor data. *arXiv:1603.00797*, 2016.

[123] Douglas L Vail, Manuela M Veloso, and John D Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, page 235. ACM, 2007.

[124] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[125] Aurore Voldoire, E Sanchez-Gomez, D Salas y Mélia, B Decharme, Christophe Cassou, S Sénési, Sophie Valcke, I Beau, A Alias, M Chevallier, et al. The cnrm-cm5. 1 global climate model: description and basic evaluation. *Climate Dynamics*, 40(9-10):2091–2121, 2013.

[126] Kiel von Lindenberg. Comparative analysis of gps data. *Undergraduate Journal of Mathematical Modeling: One+ Two*, 5(2):1, 2014.

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

[128] Shuangquan Wang, Jie Yang, Ningjiang Chen, Xin Chen, and Qinfeng Zhang. Human activity recognition with user-free accelerometers in the sensor networks. In *Proceedings of the International Conference on Neural Networks and Brain*, volume 2, pages 1212–1217. IEEE, 2005.

[129] Xiaofeng Wang, Donald E Brown, and Matthew S Gerber. Spatio-temporal modeling of criminal incidents using geographic, demographic, and twitter-derived information. In *Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on*, pages 36–41. IEEE, 2012.

[130] A Weiss, T Herman, M Plotnik, M Brozgol, N Giladi, and JM Hausdorff. An instrumented timed up and go: the added value of an accelerometer for identifying fall risk in idiopathic fallers. *Physiological measurement*, 32(12):2003, 2011.

[131] Mark William Woolrich, Mark Jenkinson, J Michael Brady, and Stephen M Smith. Fully bayesian spatio-temporal modeling of fmri data. *IEEE transactions on medical imaging*, 23(2):213–231, 2004.

[132] Elizabeth Wu, Wei Liu, and Sanjay Chawla. Spatio-temporal outlier detection in precipitation data. In *International Workshop on Knowledge Discovery from Sensor Data*, pages 115–133. Springer, 2008.

[133] Yangyang Xie, Bin Zhao, Lin Zhang, and Rong Luo. Spatiotemporal variations of pm2. 5 and pm10 concentrations between 31 chinese cities and their relationships with so2, no2, co and o3. *Particuology*, 20:141–149, 2015.

[134] Jianpeng Xu, Xi Liu, Tyler Wilson, Pang-Ning Tan, Pouyan Hatami, and Lifeng Luo. Muscat: Multi-scale spatio-temporal learning with application to climate modeling. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI'18)*, pages 2912–2918, 2018.

[135] Jianpeng Xu, Pang-Ning Tan, and Lifeng Luo. Orion: Online regularized multi-task regression and its application to ensemble forecasting. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 1061–1066. IEEE, 2014.

[136] Jianpeng Xu, Pang-Ning Tan, Lifeng Luo, and Jiayu Zhou. Gspartan: a geospatio-temporal multi-task learning framework for multi-location prediction. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 657–665. SIAM, 2016.

[137] Jianpeng Xu, Pang-Ning Tan, Jiayu Zhou, and Lifeng Luo. Online multi-task learning framework for ensemble forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1268–1280, 2017.

[138] Jianpeng Xu, Jiayu Zhou, and Pang-Ning Tan. Formula: Factorized multi-task learning for task discovery in personalized medical models. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 496–504. SIAM, 2015.

[139] Jianpeng Xu, Jiayu Zhou, Pang-Ning Tan, Xi Liu, and Lifeng Luo. Wisdom: Weighted incremental spatio-temporal multi-task learning via tensor decomposition. In *IEEE International Conference on Big Data (Big Data'2016)*, pages 522–531. IEEE, 2016.

[140] Yan Yan, Elisa Ricci, Gaowen Liu, and Nicu Sebe. Egocentric daily activity recognition via multitask clustering. *IEEE Transactions on Image Processing*, 24(10), 2015.

[141] Haiqin Yang, Irwin King, and Michael R Lyu. Online learning for multi-task feature selection. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1693–1696. ACM, 2010.

[142] Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y Chai. Grounded semantic role labeling. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 149–159, 2016.

[143] Yongxin Yang and Timothy M Hospedales. Trace norm regularised deep multi-task learning. *arXiv preprint arXiv:1606.04038*, 2016.

[144] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[145] Lei Yuan, Yalin Wang, Paul M Thompson, Vaibhav A Narayan, and Jieping Ye. Multi-source learning for joint analysis of incomplete multi-modality neuroimaging data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1149–1157. ACM, 2012.

[146] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.

[147] Liang Zhao, Qian Sun, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. Multi-task learning for spatio-temporal event forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1503–1512. ACM, 2015.

[148] Jiayu Zhou, Jianhui Chen, and Jieping Ye. Malsar: Multi-task learning via structural regularization. *Arizona State University*, 21, 2011.

[149] Jiayu Zhou, Lei Yuan, Jun Liu, and Jieping Ye. A multi-task learning formulation for predicting disease progression. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 814–822. ACM, 2011.

[150] Xun Zhou, Shashi Shekhar, and Reem Y Ali. Spatiotemporal change footprint pattern discovery: an inter-disciplinary survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(1):1–23, 2014.