DEEP CONVOLUTIONAL NETWORKS FOR MODELING GEO-SPATIO-TEMPORAL RELATIONSHIPS AND EXTREMES

By

Tyler Wilson

A DISSERTATION

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

Computer Science – Doctor of Philosophy

2021

ABSTRACT

DEEP CONVOLUTIONAL NETWORKS FOR MODELING GEO-SPATIO-TEMPORAL RELATIONSHIPS AND EXTREMES

By

Tyler Wilson

Geo-spatio-temporal data are valuable for a broad range of applications including traffic forecasting, weather prediction, detection of epidemic outbreaks, and crime monitoring. Data driven approaches to these problems must address several fundamental challenges such as handling the geo-spatio-temporal relationships and extreme events. Another recent technological shift has been the success of deep learning especially in applications such as computer vision, speech recognition, and natural language processing. In this work, we argue that deep learning is a promising approach for many geo-spatio-temporal problems and highlight how it can be used to address the challenges of modeling geo-spatio-temporal relationships and extremes. Though previous research has established techniques for modeling spatio-temporal relationships, these approaches are often limited to gridded spatial data with fixed-length feature vectors and considered only spatial relationships among the features, while ignoring the relationships among model parameters.

We begin by describing how the spatial and temporal relationships for non-gridded spatial data can be modeled simultaneously by coupling the graph convolutional network with a long short-term memory (LSTM) network. Unlike previous research, our framework treats the adjacency matrix associated with the spatial data as a model parameter that can be learned from data, with constraints on its sparsity and rank to reduce the number of estimated parameters. Further, we show that the learned adjacency matrix may reveal useful information about the dominant spatial relationships that exist within the data. Second, we explore the varieties of spatial relationships that may exist in a geo-spatial prediction task. Specifically, we distinguish between spatial relationships among predictors and the spatial relationships among model parameters at different locations. We demonstrate an approach for modeling spatial dependencies among model parameters using graph convolution and provide guidance on when convolution of each type can be effectively applied. We

evaluate our proposed approach on a climate downscaling and weather prediction tasks. Next, we introduce DeepGPD, a novel deep learning framework for predicting the distribution of geo-spatio-temporal extreme events. We draw on research in extreme value theory and use the generalized Pareto distribution (GPD) to model the distribution of excesses over a threshold. The GPD is integrated into our deep learning framework to learn the distribution of future excess values while incorporating the geo-spatio-temporal relationships present in the data. This requires a novel reparameterization of the GPD to ensure that its constraints are satisfied by the outputs of the neural network. We demonstrate the effectiveness of our proposed approach on a real-world precipitation data set. DeepGPD also employs a deep set architecture to handle the variable-sized feature sets corresponding to excess values from previous time steps as its predictors. Finally, we extend the DeepGPD formulation to simultaneously predict the distribution of extreme events and accurately infer their point estimates. Doing so requires modeling the full distribution of the data not just its extreme values. We propose DEMM, a deep mixture model for modeling the distribution of both excess and non-excess values. To ensure the point estimation of DEMM is a feasible value, new constraints on the output of the neural network are introduced, which requires a new reparameterization of the model parameters of the GPD. We conclude by discussing possibilities for further research at the intersection of deep learning and geo-spatio-temporal data.

ACKNOWLEDGEMENTS

Looking back over my time at Michigan State is surreal. I never could anticipated the what would be required for me to see my PhD program to completion or all the ups and downs (and global pandemics) along my journey. Reaching this point would have been impossible without help from colleagues, friends, and family.

I would like to begin by extending special thanks to my adviser, Dr. Pang-Ning Tan. When I entered graduate school I had no understanding of research and I could not have asked for a better guide. Enumerating all that I have learned from him would be an impossible task but I will always remember his emphasis on storytelling in communication and his strong work ethic. He has consistently offered patient advice, direction, and insight. Having him as a mentor has been an incredible privilege.

Dr. Lifeng Luo has been a similarly valuable mentor. His deep domain knowledge and warm support were invaluable. I would also like to thank my other committee members Dr. Jiliang Tang and Dr. Jiayu Zhou for their valuable feedback on my research.

In addition, I would like to thank my lab mates Jianpeng Xu, Shuai Yuan, Courtland VanDam, Ding Wang, Farzan Masrour, Pouyan Hatami, Boyang Liu, Asadullah Galib, Francisco Santos, and Anna Stephens for their friendship and collaboration. Having them along with me on this journey has deeply enriched my time in graduate school.

When times were difficult and I was feeling stressed I was always glad to have friends who were totally disconnected from the world of computer science research including Brenda Kronemeijer Heyink, Matthijs Kronemeijer, Elizabeth Clarke, Chris Rich, Tommy Winberry, Shaun Anderson, John Gause, Jake Porter, Sam Mimms, and Kevin Mimms that I could turn to.

And finally, I would like to thank my parents, Susan and Larry Wilson, who have always loved me; my brother Chris for being a constant source of fun and positivity; and my sister Laura who I love deeply even if she couldn't quite beat me to a doctorate.

This dissertation was partially supported by the National Science Foundation under grants IIS-

2006633 and IIS-1615612 as well as Michigan State University through computational resources provided by the Institute for Cyber Enabled Research.

TABLE OF CONTENTS

LIST OF TABLES x

LIST OF FIGURES xi

CHAPTER 1 INTRODUCTION 1
1.1 Challenges 1
1.2 Deep Learning for Geo-spatio-temporal Data 3
1.3 Thesis Contributions 6
1.3.1 Modeling Geo-Spatio-Temporal Data with Hybrid Graph Convolution and LSTM 6
1.3.2 Modeling Spatial Relationships among Model Parameters using Convolution 7
1.3.3 Prediction of Tail Distribution in Geo-Spatio-Temporal Data 7
1.3.4 Joint Modeling of Tail Distribution and Point Prediction of Geo-Spatio-Temporal Data 8
1.3.5 Thesis Organization 9
1.3.6 Publications 9

CHAPTER 2 RELATED WORK 11
2.1 Deep Learning for Spatio-temporal Data 11
2.2 Deep Learning For Geo-spatio-temporal Data 13

CHAPTER 3 A LOW RANK WEIGHTED GRAPH CONVOLUTIONAL LSTM APPROACH FOR MODELING GEO-SPATIO-TEMPORAL DATA 16
3.1 Introduction 16
3.2 Preliminaries 19
3.2.1 Long Short-Term Memory Networks 20
3.2.2 Convolutional Neural Networks 21
3.3 Proposed Approach 22
3.3.1 Graph Convolution 22
3.3.2 Graph Convolutional LSTM 23
3.3.3 Weighted Graph Convolutional LSTM 24
3.4 Experimental Evaluation 26
3.4.1 Weather Datasets 26
3.4.2 Baseline Algorithms 27
3.4.3 Experimental Setup 28
3.4.4 Experimental Results 29
3.4.4.1 Comparison against Baseline Methods 30
3.4.5 Low-Rank versus Sparse WGC-LSTM 31
3.4.6 Number of Layers 32
3.4.7 Learned Adjacency Matrix Visualization 32
3.5 Related Work 35

3.5.1 Deep Learning . . . 35
3.5.2 Graph Convolution . . . 37
3.5.3 Weather Prediction . . . 38
3.6 Conclusions and Future Work . . . 38

CHAPTER 4 CONVOLUTIONAL METHODS FOR PREDICTIVE MODELING OF GEOSPATIAL DATA . . . 40
4.1 Introduction . . . 40
4.2 Related Work . . . 42
4.3 Convolutional Methods for Modeling Geospatial Data . . . 42
4.3.1 Convolution on Input ($Conv_x$) . . . 43
4.3.2 Convolution on Parameter ($Conv_w$) . . . 45
4.3.3 Hybrid Convolution Approach ($Conv_{xw}$) . . . 46
4.4 Choice of Convolutional Method . . . 47
4.4.1 When $Conv_x$ is useful? . . . 47
4.4.2 When $Conv_w$ is useful? . . . 48
4.4.3 When $Conv_{xw}$ is useful? . . . 48
4.5 Experimental Evaluation . . . 49
4.5.1 Experiments on Synthetic Data . . . 49
4.5.1.1 Experimental Results on Synthetic Data . . . 51
4.5.2 Experiments on Real-world Data . . . 51
4.5.2.1 Experimental Setup . . . 52
4.5.2.2 Experimental Results . . . 52
4.5.2.3 Ablation Study . . . 54
4.5.2.4 Embedding Visualization . . . 54
4.5.2.5 Discussion . . . 56
4.6 Conclusion . . . 57

CHAPTER 5 A DEEP LEARNING APPROACH FOR MODELING GEOSPATIO-TEMPORAL EXTREME EVENTS . . . 58
5.1 Introduction . . . 58
5.2 Related Work . . . 60
5.3 Preliminaries . . . 61
5.3.1 Extreme Value Theory . . . 62
5.3.2 Deep Set . . . 63
5.4 Proposed DeepGPD Framework . . . 64
5.4.1 Local Feature Extraction . . . 64
5.4.2 Spatial Feature Extraction . . . 65
5.4.3 Extreme Value Modeling (EVM) . . . 66
5.5 Experimental Results . . . 68
5.5.1 Comparison against Baseline Methods . . . 71
5.5.2 Distribution of Estimated GPD Parameters . . . 73
5.6 Conclusions . . . 75

CHAPTER 6 DEEP EXTREME MIXTURE MODEL FOR GEO-SPATIO-TEMPORAL DATA 76
6.1 Introduction 76
6.2 Related Work 78
6.3 Preliminaries 79
6.3.1 Problem Statement 79
6.3.2 Extreme Value Theory 80
6.3.3 Hurdle Model 81
6.4 Deep Extreme Mixture Model 82
6.4.1 Mixture Model 82
6.4.2 Deep Neural Network 85
6.4.3 Constraint Enforcement 86
6.4.3.1 GPD Constraints 87
6.4.4 Training 88
6.5 Experimental Evaluation 89
6.5.1 Data 89
6.5.2 Baseline Methods 89
6.5.3 Experimental Setup 90
6.5.4 Experimental Results 91
6.5.4.1 Performance Comparison against Baseline Methods 91
6.5.4.2 Ablation Studies 93
6.5.4.3 Spatial Analysis 93
6.5.4.4 Extreme Event Frequency 94
6.5.4.5 Extreme Event Timing 96
6.6 Conclusion 96

CHAPTER 7 CONCLUSION AND FUTURE WORK 98
7.1 Future Work 99
7.1.1 Spatio-temporal Relationships 99
7.1.2 Extreme Events 100

BIBLIOGRAPHY 102

LIST OF TABLES

Table 3.1: Summary of weather datasets. 27

Table 3.2: Comparison of R^2 value for graph convolutional LSTM against other baseline methods. 29

Table 3.3: Percent of stations for which the model on each row outperforms the model in the column on the temperature prediction task for the GSOD dataset. 29

Table 3.4: Percent of stations for which the model on each row outperforms the model in the column on the wind speed prediction task for the GSOD dataset. 29

Table 4.1: R-squared results for synthetic data. 49

Table 4.2: Summary of real-world datasets. 50

Table 4.3: Comparison of convolutional methods against other baselines using R-squared as evaluation metric. 50

Table 4.4: Comparing the R-squared values of linear and non-linear convolution on inputs for GHCN data. 54

Table 4.5: Correlation between regression weights and local embeddings with geographic distance. 55

Table 5.1: Comparison between `DeepGPD` against baseline methods in terms of negative log-likelihood (NLL) and correlation (ρ) of predicted ξ and σ to ground truth values. 69

Table 5.2: Results of ablation study. 70

Table 6.1: Predictive performance comparison 92

LIST OF FIGURES

Figure 3.1: Plot of changes in temperature at a weather station (x-axis) against (a) LSTM prediction error and (b) average temperature change at nearby locations in the previous time step. 16

Figure 3.2: Plot of changes in GCN prediction error (y-axis) against (a) temperature change and (b) temperature change at neighboring locations 17

Figure 3.3: Proposed model 19

Figure 3.4: Plot of IGRA stations with station size scaled to be proportional to the cubic value of MSE of the WGC-LSTM for temperature prediction at that station. We see that there are larger errors in the north. Correlation between a station's average error and station temperature standard deviation is 0.82. 31

Figure 3.5: Comparison between low rank WGC-LSTM and sparse WGC-LSTM. 32

Figure 3.6: Prediction error for wind speed as a function of the number of layers in WGC-LSTM. 33

Figure 3.7: Prediction error for wind speed as a function of the number of layers in WGC-LSTM. 33

Figure 3.8: Comparison between the sparse and low rank adjacency matrices learned for temperature prediction on the IGRA dataset. 34

Figure 3.9: Comparing the results of clustering performed on the sparse and low rank adjacency matrices learned for temperature prediction on the IGRA dataset are used for clustering. An adjacency matrix formed from geographic distance is also clustered. 36

Figure 4.1: Three different architectures for performing convolution on geospatial data. . . . 43

Figure 4.2: Flow charts showing when different kinds of convolution are applicable 47

Figure 4.3: Proportion of stations in which $Conv_w$ beats the baseline methods as the size of the training set varies. 53

Figure 4.4: Relationship between location embeddings, regression weights, and geographic distance. 55

Figure 4.5: Performance comparison between convolutional and baseline methods on the real-world datasets. 57

Figure 5.1: Relationship between shape parameter ξ of generalized Pareto distribution for modeling precipitation excesses in two successive time windows. 60

Figure 5.2: Proposed `DeepGPD` framework. 64

Figure 5.3: Fitted p-Value distribution of K-S test. 67

Figure 5.4: Mean residual life plot for excess precipitation. 68

Figure 5.5: Relationship between predictive improvements over baselines and true ξ. . . . 71

Figure 5.6: Comparison between the spatial distribution of the true and predicted ξ values for linear GPD and `DeepGPD`. 72

Figure 6.1: An overview of the proposed *DEMM* architecture 82

Figure 6.2: Plots showing the improvement of the *DEMM* over a hurdle model as the amount of rainfall varies. Positive values indicate samples where the *DEMM* has achieves a lower loss than hurdle model baseline. 92

Figure 6.3: Plot showing how the negative log-likelihood varies for both the *DEMM* and the fixed threshold variant of the *DEMM*. Shaded region represents ± 1 standard deviation. 93

Figure 6.4: Plots showing the spatial locations where the *DEMM* outperforms the hurdle model in terms of their mean absolute error (MAE). Red indicates that the *DEMM* is outperforming. 94

Figure 6.5: The relationship between average precipitation at a location and the margin by which the *DEMM* outperforms the ensemble mean at that location. Positive values indicate the *DEMM* outperforms. 95

Figure 6.6: Plot showing how the brier score for the extreme events class varies with the quantile used to define extreme events. 95

Figure 6.7: Plot comparing the observed frequency of extreme events (x-axis) against the predicted frequency of extreme events. 96

CHAPTER 1

INTRODUCTION

In machine learning and data mining it is important for the algorithms we use to be suited for the kind of data we are working with. One major class of data is geo-spatio-temporal data. As the name suggests, this is geographic data where the spatial and temporal relationships between samples are important. For example, spatio-temporal relationships are crucial for predicting the day to day variations of weather as well as predicting the the behavior of extreme weather phenomena like the track and intensity of a hurricane. Climatology has many examples of geo-spatio-temporal tasks including climate downscaling, where the goal is to obtain future projections of the climate at a fine scale from coarse scale forecasts generated by global or regional climate models. Other important geo-spatio-temporal data sets include traffic, air quality, streamflow, and crime data. The broad range of problems that fall under the umbrella of geo-spatio-temporal prediction make them an important subject of study.

This chapter will be organized as follows: We begin by describing the key challenges of geo-spatio-temporal data that will serve as the primary focus of this dissertation. Next, we will briefly introduce deep learning and convolutional neural networks in Section 1.2 and motivate their application to geo-spatio-temporal data. Finally, in Section 1.3, we will describe the contributions of this thesis and describe its overall organization.

1.1 Challenges

Geo-spatio-temporal data has a number of challenges that must be addressed in order for predictive modeling techniques to be fruitfully applied. In this dissertation we focus primarily on two of these challenges, namely, how to handle the geo-spatio-temporal relationships and extreme events.

One of the most important challenges when working with geo-spatio-temporal data is to accurately model the relationships that exist within the data. According to Tobler's first law of geography "[E]verything is related to everything else, but near things are more related than distant things"

[88]. In many geographic applications, the spatial dependencies may manifest itself in various ways, as spatial relationships among predictors, spatial relationships between predictors and the target variable, or spatial relationships between model parameters at different locations. The latter is evidenced by the successful application of multi-task learning methods to various geospatial prediction problems [66, 105]. The challenge here is to identify the type of spatial dependencies that can be exploited to improve the performance of predictive models in a given application. In particular, though geographic distance is commonly used to characterize geo-spatial relationships, other factors often come into play depending on the application. For example, in weather prediction, prevailing winds may have a significant impact on the relationship between two locations. Temporal dependencies are another important aspect of geo-spatio-temporal relationships with factors such as periodicity and autocorrelation that must be taken into consideration. A standard assumption in machine learning is that data is independently and identically distributed (i.i.d.). As a result, many common machine learning techniques will be inappropriate when applied to geo-spatio-temporal data due to the importance of spatial and temporal autocorrelations. Another significant challenge is that the geo-spatio-temporal relationships is not necessarily linear. Weather, for instance, is a well known example of a non-linear system. Utilizing models that are capable of learning these non-linear relationships will therefore be crucial.

The modeling of extreme events is the second major challenge to be investigated in this dissertation. Many important phenomena in geo-spatio-temporal applications involve extreme events. As an example, though accurate prediction of precipitation in any amount is valuable, extreme precipitation is especially important as it can cause flooding, which leads to severe property damage and potentially the loss of human lives. Thus, models that accurately predict modest amount of rainfall, e.g., scattered showers that occur in a given day, may not be as useful to stakeholders responsible for planning municipal responses to flooding compared to models that accurately predict when large amounts of rain will fall in a short time. This is a challenge since many predictive models are designed to predict the conditional mean of a probability distribution and their predictions will thus tend to underestimate the magnitude of extreme events which lie in the upper tail of the

probability distribution being modeled. While there exists substantial research in the application of linear models for modeling extreme values [22], these approaches often fail to fully account for the complex non-linear relationships inherent in many geo-spatio-temporal applications. The model for predicting extreme events must effectively capture the tail distribution of the forecasted variable as well as the magnitude, frequency, and timing of the extreme events.

1.2 Deep Learning for Geo-spatio-temporal Data

Deep learning has gained wide-spread popularity particularly in applications such as computer vision and natural language processing over the last decade. The basic idea is to create complex, highly non-linear, over-paramaterized functions through the composition of relatively simple functions like affine transformations. An efficient approach to gradient descent called "back-propagation" can then be used to optimize the parameters of the deep learning model. We argue that deep learning is a promising tool for geo-spatio-temporal applications due to its demonstrated capacity for learning highly complex, non-linear feature representation of the data.

Some of the most successful applications of deep learning have been to tasks with spatial and temporal/sequential relationships. Computer vision problems, where deep learning has been applied, require effective ways of modeling spatial relationships between pixels of an image. Similarly, in natural language processing, modeling the sequential relationships between tokens in text corpus is of the utmost importance and this is closely related to temporal modeling. Video data is an example application of deep learning to spatio-temporal data in computer vision. More recently, deep learning has also found success in a growing number of weather and climate prediction problems [26, 66, 77, 80]. All this suggests that there is a strong further potential for deep learning to be applied to geo-spatio-temporal problems. A second reason that deep learning is a promising tool for working with geo-spatio-temporal data is that deep learning models are capable of learning complex non-linear functions. In addition to empirical results demonstrating the effectiveness of deep learning on highly non-linear problems, there are theoretical results establishing that deep learning models are universal function approximators [41, 116]. Together,

the strong track record of deep learning in spatial and sequential applications and the effectiveness of deep learning in modeling non-linear relationships makes it a promising technique for geo-spatio-temporal applications.

Convolutional neural networks is arguably one of the most popular deep learning models for spatio-temporal applications. Standard convolutional neural networks can be used for modeling spatial relationships [53], temporal/sequential relationships [5], and spatio-temporal relationships [80] as long as the data has a gridded structure. For instance, images can be represented as grids of pixels whereas time series with observations made at regular intervals can be treated as 1-dimensional gridded data sets. The central component of all convolutional neural networks is the convolutional layer. In a 2-d spatial context, each convolutional layer consists of a collection of small 2-dimensional grids of model parameters called filters. These filters compute the weighted sums of small localized regions of the input grid. We can formally express the convolution operation for a 2-d grid as follows:

$$y_{i,j} = \sum_{m=-\infty}^{k} \sum_{n=-\infty}^{k} f_{m,n} x_{i-m,j-n} \tag{1.1}$$

where $x_{i-m,j-n}$ is the element in row $i-m$ and column $j-n$ of the input 2-d grid, $y_{i,j}$ is the element in row i and column j of the grid output by the convolution operation, $f_{m,n}$ is the element in the m row and n column of a $k \times k$ filter. By composing many of these simple convolution operations, interleaved with other non-linear mathematical operations, it is possible to build convolutional neural networks capable of modeling complex non-linear spatio-temporal relationships.

However, the geo-spatio-temporal modeling tasks raise some additional challenges for deep learning. As described above, the traditional way of modeling spatial relationships with deep learning is to use a convolutional neural network. This approach works well for images, where the pixels are arranged into a regular grid. For geo-spatio-temporal data that is similar to image data (e.g. satellite images or the output of climate model simulations with regular grids) these traditional deep learning approaches will provide a good starting point. However, in other cases, the spatial locations may not reside on a regularly spaced grid. For example, weather observations are often made at weather stations scattered irregularly so modeling spatial relationships using ordinary

gridded 2-d convolution is not an option. Furthermore, convolutional neural network is typically used to capture spatial relationships among the input features. Applying convolutional neural network to capture spatial relationship among model parameters, in a similar fashion as multi-task learning, is an issue that has not been sufficiently explored in the literature. In addition, current deep learning architectures are mostly designed for modeling typical scenarios instead of extreme phenomena. Most of the research studying extreme values in data is done by statisticians using relatively simple models [22]. Adapting deep learning architectures to extreme value prediction remains an important but under-studied research area.

Interpretability is often a desirable property for a machine learning model. Though deep learning models can achieve strong predictive performance, this is often inadequate if the resulting model is not interpretable. Interpretability is important in part for demonstrating the reliability of the learned model since users are more likely to trust a model that they can understand [75]. A model that is uninterpretable but performs well on a given data set may simply capture spurious patterns that do not generalize well to the rest of the sample space from which the data set was drawn. By developing interpretable models, this allows for the domain experts to verify their reliability. Interpretability is also key to helping domain experts understand the justification behind the model's predictions. By better understanding why the model is able to make high quality predictions, there is a chance that the domain experts may develop new insights or generate new hypothesis to test the phenomenon being studied.

Another challenge in applying deep learning to geo-spatio-temporal data is that in many applications the available data is inherently limited. Monthly climate data for a given location is generated at a rate of 12 samples per year and there is nothing that can be done to increase this rate. When dealing with the extreme events this problem is further exacerbated because extreme events are, by definition, rare. Though this dearth of data will affect any data-driven approach, it is especially worrisome when using deep learning since deep learning requires large amounts of data to ensure adequate training and prevent model overfitting, given the large number of parameters to be estimated from the training data.

1.3 Thesis Contributions

This thesis focuses on the development of novel deep learning frameworks to address the challenges described in the previous section, with particular attention paid to the ways deep learning can help to model geo-spatio-temporal relationships and extreme values. Specifically, Chapters 3 and 4 will focus primarily on how deep learning can be used to model geo-spatio-temporal relationships while Chapters 5 and 6 examine how deep learning can be used to predict extreme events in geo-spatio-temporal data. A detailed summary of the thesis contributions is given in the subsections below.

1.3.1 Modeling Geo-Spatio-Temporal Data with Hybrid Graph Convolution and LSTM

Chapter 3 proposes a deep learning architecture called the weighted graph convolutional LSTM (WGC-LSTM) for geo-spatio-temporal prediction. At a high level, the goal of WGC-LSTM is to take a sequence of observations of variables at a fixed set of locations and accurately predict the value of one of those variables in the next time step. In order to make these predictions the WGC-LSTM must model geo-spatio-temporal relationships but doing so requires addressing several challenges. First, the relationships between locations where variables are observed are difficult to characterize because the strength of relationships may be influenced by factors other than distance. Second, variables are observed at irregularly spaced locations and not on a grid so modeling spatial relationships using ordinary convolution is not an option. To address these difficulties, a graph structure is imposed on the data, where different locations correspond to vertices and their relationships correspond to edges. A graph convolutional LSTM is developed to generate the spatio-temporal predictions. An adjacency matrix is necessary for the formulation of graph convolution but rather than computing the entries of the adjacency matrix based on distance as pre-processing step, we instead treat its entries as model parameters so that the relationships between different locations can be inferred in a purely data-driven way. However, learning relationships in this way can potentially introduce a number of parameters that is quadratic in the number of

locations so we propose two different techniques for reducing overfitting. When applied to the weather prediction problem, visualization of the resulting learned adjacency matrix shows that they are consistent with climatological science, which suggests that the spatial relationships extracted by WGC-LSTM may provide interpretable insights into the data.

1.3.2 Modeling Spatial Relationships among Model Parameters using Convolution

As previously established in Section 1.1, modeling spatial relationships is an important challenge in geo-spatio-temporal applications, but typically only spatial relationships between predictor variables are considered. However, if each location is assigned its own predictive model, then it is possible to consider spatial relationships between the model parameters at different locations as well. The potential of modeling spatial relationships among model parameters has previously been demonstrated in the context of multi-task learning [66, 105], and the effectiveness of graph convolution to model spatial relationships among predictors using WGC-LSTM raises the possibility that it could be used to model spatial relationships among parameters as well. In Chapter 4 we consider two research questions: First, can graph convolution be used to model spatial relationships among model parameters, and if so, how? And second, when should we model spatial relationships among model parameters and when should we model spatial relationships among predictors? In Chapter 4 we answer these questions by proposing a graph convolution based framework capable of modeling spatial relationships amond predictors, parameters, or both. In addition, we also provide and validate advice on when modeling different spatial relationships will be useful.

1.3.3 Prediction of Tail Distribution in Geo-Spatio-Temporal Data

Predictive modeling of extreme events is crucial for many geo-spatio-temporal applications. In Chapter 5, a novel framework called *DeepGPD* is presented to predict the tail distribution of a target variable (e.g., precipitation) for a future time period. The distribution can be used to infer various statistics about the future occurrence of extreme events such as their expected values, quantiles, moments, and return period [21]. This is accomplished using a neural network to

infer the parameters of the generalized Pareto distribution. The generalized Pareto distribution is one of the most frequently utilized distributions within extreme value theory and governs the distribution of excesses above a fixed threshold. The conjunction of deep learning and extreme value theory has not been extensively studied and introduces some challenges – particularly around the enforcement of constraints on extreme value distribution parameters. DeepGPD employs a novel reparameterization trick to enforce these constraints. In addition, another challenge related to extreme value modeling with deep learning is how to best utilize historical extreme events as predictors. Conventional deep learning methods typically use vector-valued predictors. Since the number of extreme events can vary over time, using historical extreme events as predictors requires a way of modeling set-valued predictors. The proposed *DeepGPD* framework incorporates deep set [111] to transform the set-valued predictors into fixed length vectors before they are integrated into the learning algorithm.

1.3.4 Joint Modeling of Tail Distribution and Point Prediction of Geo-Spatio-Temporal Data

While knowing the future distribution of extreme events is useful, particularly for long-term forecasts, there are numerous applications that require accurate point prediction at each future time step (e.g., for short-term weather forecasting). However, making point predictions of a target variable conditioned on the predictors cannot be accomplished based solely on the distribution of extreme values. It requires knowledge of the entire conditional distribution of the target variable—both the extreme and non-extreme values. In addition, when modeling extreme values as excesses above a threshold, the choice of threshold effectively defines what should be considered an extreme value. Unfortunately, finding the appropriate threshold for extreme events may vary depending on the application and user requirement. This may result in scenarios where users may have a strong predictive model but need to retrain it from scratch if they wish to consider another threshold. The final contribution of this thesis is to extend the *DeepGPD* framework to account for these challenges. Specifically, *DeepGPD* is extended to a mixture model formulation called *DEMM* with the generalized Pareto distribution used as one of its components, thus enabling it to predict the

full conditional distribution of the target variable. This full conditional distribution can then be used as the basis for making point predictions by using the expected value (mean) of the mixture model. However, this requires computing the mean of each of its components, including the mean of the generalized Pareto distribution. Since the mean of the generalized Pareto distribution is only defined when its shape parameter is less than 1, additional constraints on the generalized Pareto distribution parameters are needed beyond those required for the *DeepGPD*. In addition, *DEMM* also provides a method for altering the threshold defining extreme events at test time without having to retrain the model.

1.3.5 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 reviews the background literature and presents the relevant previous works related to this research. Chapter 3 introduces the WGC-LSTM framework for modeling geo-spatio-temporal data, with application to weather prediction. Chapter 4 investigates the effectiveness of using of convolutional neural networks to capture different aspects of spatial relationships, including dependencies among the features and the model parameters. In Chapter 5, we will describe the DeepGPD framework for predicting the distribution of extreme events, while Chapter 6 extends the framework to support point predictions, Finally, Chapter 7 presents the conclusions and directions for future work.

1.3.6 Publications

Some content in this dissertation was adapted from the following works:

- Tyler Wilson, Pang-Ning Tan, and Lifeng Luo. A Low Rank Weighted Graph Convolutional Approach to Weather Prediction. In Proceedings of IEEE International Conference on Data Mining (ICDM 2018), Singapore (2018).
- Tyler Wilson, Pang-Ning Tan, and Lifeng Luo, "Convolutional Methods for Predictive Modeling of Geospatial Data". In Proceedings of the SIAM International Conference on Data

Mining (SDM-2020), Cincinnati, OH (2020)

- Tyler Wilson, Pang-Ning Tan, Lifeng Luo. "DeepGPD: A Deep Learning Approach for Modeling Geospatio-Temporal Extreme Events". To appear in Proceedings of 36th AAAI Conference on Artificial Intelligence (AAAI-2022), Vancouver, Canada (2022).

- Tyler Wilson, Pang-Ning Tan, Lifeng Luo, Andrew McDonald, Asadullah Galib. "DEMM: Deep Extreme Mixture Model", in preparation (2022).

When a dissertation chapter is adapted from prior work this will be indicated with a footnote in that chapter's introduction.

CHAPTER 2

RELATED WORK

In this section we describe the current state of research in the application of deep learning to geo-spatio-temporal data. Some of the most common applications domains of deep learning to geo-spatio-temporal data include to weather forecasting [36], precipitation nowcasting [80] and precipitation estimation [85, 86, 87], remote sensing [46, 68, 82, 113], climate downscaling [94], and traffic forecasting [61, 62, 73, 106, 108, 110].

One of the most important aspects of geo-spatio-temporal data is modeling the spatio-temporal relationships and when deep learning is applied to geo-spatio-temporal data one of its most frequent uses is to model spatio-temporal relationships. The way this is accomplished will be the focus of our discussion here. We will start with a brief overview of research into deep learning with special attention paid to the basic deep learning architectures used to model spatial, temporal, and spatio-temporal relationships. We will then describe how these techniques are used and adapted for geo-spatio-temporal data.

2.1 Deep Learning for Spatio-temporal Data

Deep learning [58] has gained considerable attention over the last several years due to its success in artificial intelligence applications. Krizhevsky et al. [53] used a convolutional neural network [59] to achieve state of the art performance on a large object recognition data set. After sparking the resurgent interest in deep learning, convolutional neural networks have become one of the most significant objects of study in the research and application of deep learning. The early success of convolutional neural networks has been built upon with general purpose architectural changes [38, 81, 84] as well as the development of new architecture to support a broader range of applications including object detection [32], semantic segmentation [76], image generation [34], and pose estimation [89].

The strength of convolutional neural networks lies in their ability to model the spatial relationship

between the pixels of an image. We expect that nearby pixels in an image will be more closely related to one another than pixels that are distant – a phenomenon known as "spatial auto-correlation". A similar intuition motivates the application of convolutional neural networks to sequential and temporal data where we use convolutional neural networks to model temporal auto-correlation. Among the earliest instances of what we now call a convolutional neural network was a "time delay neural network" applied to the problem of phoneme recognition [99]. Like modern deep learning models it utilized 1-d convolution to model temporal relationships and was trained using backpropagation. The use of convolutional neural networks to model temporal relationships has continued to the present and recently in Bai et al. [5] the authors provide empirical evidence that a properly designed convolutional architecture can outperform recurrent neural networks on tasks which are often believed to be best suited for recurrent neural networks. Convolutional neural networks have recently been used to model temporal relationships in applications including audio generation [72], language modeling [45], and activity recognition in video.

Another family of deep learning models frequently used to model relationships in temporal and sequential data is recurrent neural networks (RNNs). RNNs consist of recurrent cells with an internal memory that is modified at each step in a time series using shared weights. Early formulations of the recurrent neural network suffered from an exploding and vanishing gradient problem which made them difficult to train. Hochreiter and Schmidhuber [40] addressed this problem by introducing gates that regulate the modification of the internal memory and allow the model to "remember" inputs from many time steps in the past. A variant of the long short-term memory network called a "gated recurrent unit" [20] has also gained some popularity.

In recent work, modeling spatial and temporal relationships simultaneously is typically accomplished by using a convolutional neural network for modeling the spatial relationship and either a recurrent neural network or convolutional neural network for modeling the temporal relationships. An early example of using convolution to model both spatial and temporal relationships is Chen et al. [16] which proposes a spatio-temporal deep belief network consisting of layers of convolutional boltzman machines. The layers of their network alternate between layers performing spatial con-

volution and those performing temporal convolutions so that the spatial and temporal relationships are modeled in separate layers of their model. Alternatively, Tran et al. [90] and Ji et al. [43] use 3d convolution is used to model both spatial and temporal relationships simultaneously within each layer of their network. While Tran et al. [93] use 3d convolution in lower layers to capture features related to motion while higher layers alternate between spatial and temporal convolutions.

Rather than use convolution to model temporal relationships, some research uses recurrent layers to model temporal relationships. Donahue et al. [27] combine CNNs and RNNs in a straight-forward way. They first process each frame of a video with a convolutional neural network and then feed the output feature vector produced at each frame as input into an LSTM so that the modeling of spatial and temporal relationships happens at separate locations of the model. Shi et al. [80] models spatio-temporal relationships by replacing every instance of matrix multiplication in the typical LSTM equations with 2d convolution and then feed raw image data directly to the LSTM at each time step. This results in a more tightly coupled modeling of spatial and temporal relationships since each LSTM layer now models both spatial and temporal relationships rather than only temporal ones. Hutchison et al. [3] use a hybrid of 3d convolution and recurrent neural networks for spatio-temporal modeling. Spatio-temporal features are extracted at each time step using 3d convolution and are then fed to a long short-term memory network to capture long-term dependencies among the video frames.

2.2 Deep Learning For Geo-spatio-temporal Data

In this section we provide examples of how deep learning is used to model spatial and temporal relationships in geo-spatio-temporal data. Most research into deep learning that is concerned with spatial relationships is applied to images where the spatial relationships are the result of the relationship between grids of pixels. For this reason, in any geo-spatio-temporal dataset where the data is arranged into a 2d spatial grid the same techniques, namely convolutional neural networks, can be used to model the spatial relationships [46, 57, 68, 82, 94, 113].

In some cases geo-spatial data is not arranged on an orderly spatial grid. Weather and traffic

data, for example, are often measured at stations scattered irregularly so their spatial structure is unsuitable to be input to convolutional neural networks. Though these irregularly spaced location lack a gridded structure, they can be regarded as graphs where spatial locations correspond to vertices and their spatial relationships correspond to edges. In light of this observation, researchers have drawn upon recent developments into generalizing convolutional neural networks to graphs. Generalizations of convolution are formulated either in the spatial domain or the spectral domain. Just as ordinary gridded convolution computes weighted sums of local regions of images, spatial domain generalizations of convolution to graphs computes weighted sums of neighborhoods of graphs. Prominent examples of spatial domain formulations of graph convolution include message passing neural networks [31], diffusion convolution [61], graph attention networks [97], and GraphSAGE [37]. Spectral approaches to graph convolution are theoretically well motivated approaches involving the eigendecomposition of the graph laplacian [12, 25, 51]. Though these two approaches to formulating convolution on graphs appear to be very distinct, it turns out that they share many connections. For example, in Gilmer et al. [31] the authors show that their framework for graph convolution formulated in the spatial domain actually encompasses several spectral domain formulations. There are several recent reviews of graph networks [6, 11, 102, 114] that offer a more comprehensive overview of how convolution can be generalized to graphs.

The techniques for modeling temporal relationships with deep learning in geo-spatio-temporal data are the same as those used to model temporal relationships with deep learning in other spatio-temporal problems, namely convolution [110] and RNNs [61, 108]. When attempting to model spatial and temporal relationships simultaneously, the family of techniques used is once again relatively similar to those techniques used for modeling spatio-temporal relationships with deep learning in other settings. Yu et al. [110] use convolution to model both spatial and temporal relationships while Li et al. [61] and Ye et al. [106] model spatial relationships with convolution (either gridded 2d convolution or some extension of convolution to graphs) and a recurrent neural network to model temporal relationships.

Though deep learning is most often used in the context of geo-spatio-temporal data to model

spatial and temporal relationships this is not the only way it is used. For example, in Grover et al. [36] the authors use a deep belief net to model the joint statistics of the weather variables that serve as their target resulting in a 1-2% reduction in error. In Albert et al. [2], in addition to using a CNN for to model spatial relationships the authors also use a recent architecture known as a Generative Adversarial Network [34] to ensure that generated urban land usage patterns are drawn from the same distribution as real patterns.

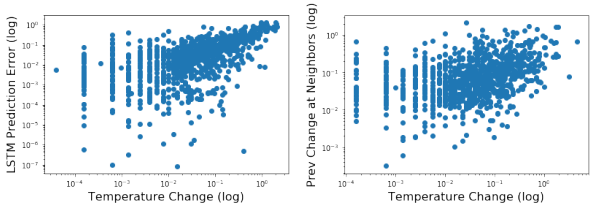
CHAPTER 3

A LOW RANK WEIGHTED GRAPH CONVOLUTIONAL LSTM APPROACH FOR MODELING GEO-SPATIO-TEMPORAL DATA

3.1 Introduction[1]

Weather prediction is an important modeling task due to its significant impact on agriculture, water resources, transportation, and many other aspects of our daily lives. Among the most popular approaches to weather prediction in use today is numerical weather prediction, which employs physics-based models to generate forecasts of future weather conditions [7]. In addition, there has also been substantial research on using data driven approaches to weather prediction [36] [69]. However, because of the nonlinearity inherent within the task, simple linear models are often insufficient to produce reliable forecasts. This has led to considerable interest in applying non-linear techniques, such as artificial neural networks [42][70] [63], to weather prediction problems. These approaches are mostly implemented using fully connected neural networks [55] [85], recurrent neural networks [112], and long short-term memory (LSTM) networks [80]. However, a major limitation of using these approaches is that they do not adequately consider the spatial autocorrelation within the dataset.

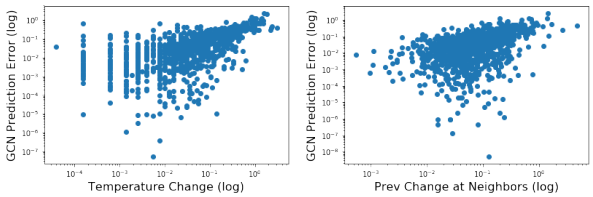
To illustrate this problem, we examine the prediction errors of an LSTM on a dataset of 12-hourly



(a) Correlation = 0.59 (b) Correlation = 0.34.

Figure 3.1: Plot of changes in temperature at a weather station (x-axis) against (a) LSTM prediction error and (b) average temperature change at nearby locations in the previous time step.

[1]**This chapter is adapted from a previously published paper: Tyler Wilson, Pang-Ning Tan, and Lifeng Luo. "A Low Rank Weighted Graph Convolutional Approach to Weather Prediction". In Proceedings of IEEE International Conference on Data Mining (ICDM 2018), Singapore (2018).**



(a) Correlation = 0.52 (b) Correlation = 0.56.

Figure 3.2: Plot of changes in GCN prediction error (y-axis) against (a) temperature change and (b) temperature change at neighboring locations

weather measurements taken at weather stations across the continental United States. LSTMs are a powerful deep learning model for time series prediction that have found success in various domains such as speech recognition [35] and machine translation [83]. However, an LSTM that makes predictions based solely on its historical weather measurements completely ignores the important spatial relationships that exist within the data. For example, Figure 3.1a compares the temperature prediction error of an LSTM at a given time step t against changes in temperature from its previous time step (i.e., $\text{Temp}_t - \text{Temp}_{t-1}$). Since the LSTM is trained for each station using its historical observations only, it tends to predict temperature values that are similar to the temperature of its previous time step, and thus, unable to anticipate rapid changes in the temperature time series. This is illustrated in Figure 3.1a, which shows the high correlation between LSTM prediction error and the temperature changes at a location. Furthermore, Figure 3.1b shows that these sudden changes tend to be preceded by large temperature changes at nearby locations in the previous time step. This observation suggests the importance of incorporating spatial information into weather prediction models. In fact, the correlation between LSTM prediction error and average temperature changes in the previous time step at nearby locations is indeed high (0.44). Thus, if the information about temperature changes in neighboring locations can be incorporated into LSTM, this may help improve its prediction accuracy.

In contrast, a convolutional neural network (CNN) is a deep learning approach for modeling spatial relationships in data. CNNs have been successfully applied to images [53], videos [47], and other spatially gridded datasets [80]. For weather prediction, as the locations of weather stations are irregularly spaced, an ordinary convolutional neural network cannot be used for modeling the data.

However, recent work has explored methods of extending convolutional neural networks so that they can be applied to arbitrary graphs [25] [51]. Just as each layer of an ordinary convolutional neural network represents each pixel as a linear combination of its neighboring pixels, a graph convolutional neural network represents each vertex in the graph as a linear combination of nearby vertices. A more detailed discussion of graph convolutional neural networks is reserved for Section 3.3. We can apply graph convolutional neural networks to the weather prediction problem by treating each weather observation location as a vertex in a graph and form edges between vertices based upon the distance between the corresponding observation locations. However, Figures 3.2a and 3.2b show that a graph convolutional neural network encounters similar type of problems as an LSTM. Whereas an LSTM is limited by its inability to consider spatial relationships, the graph convolution is unable to consider temporal relationships.

The goal of this chapter is to improve spatiotemporal prediction with deep learning by combining graph convolution with an LSTM. This can be achieved by replacing the fully connected layers within each LSTM cell with graph convolutional layers. However, care must be taken when constructing the graph to be used as input into the graph convolutional LSTM. For example, a typical approach is to consider the geographic distance between locations as edge weights of the graph. For weather prediction, the strength of the relationship between locations may also be affected by prevailing winds, topography, and other factors not directly related to distance. Consequently, distance alone may not be an optimal way to determine edge weights. To address this problem, we propose an approach to automatically learn the graph structure from data by treating the edge weights as model parameters. The learned adjacency matrix not only improves predictive performance, it also provides useful insights into how the current weather condition at a location influences the weather pattern at other locations. However, learning the full adjacency matrix introduces an additional $O(|V|^2)$ parameters, which can be computationally expensive for large graphs and may potentially lead to overfitting. To overcome these problems, we develop a technique called weighted graph convolutional LSTM (WGC-LSTM) for learning an adjacency matrix for the weather prediction problem. We propose two approaches to control overfitting in WGC-LSTM, a

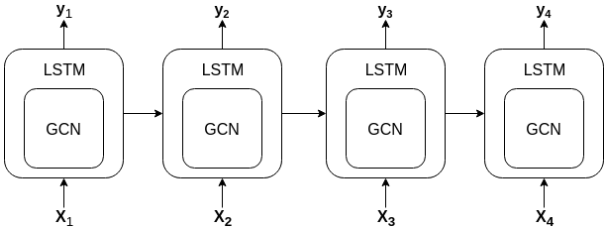


Figure 3.3: Proposed model

variant with a sparse adjacency matrix and another based on the low rank assumption. Experimental results performed on two real-world datasets demonstrate the effectiveness of the approach in terms of its predictive performance, computational efficiency, and interpretability of the learned graph. We also show that the high prediction accuracy can still be maintained but with faster runtime when using a low rank adjacency matrix for our graph convolution network.

3.2 Preliminaries

Consider a sequence of T weather observations at $|V|$ locations, i.e., $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_T$, where each $\mathbf{X}_t \in \mathbb{R}^{|V| \times d}$. The d features correspond to values of weather variables such as temperature, wind speed, and humidity from previous time steps.

A full discussion of the specific weather variables used is reserved for Section 3.4.1. Given this sequence of historical observations, the goal is to predict the target sequence $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T$, where $\mathbf{y}_t \in \mathbb{R}^{|V|}$ is a vector of the targeted weather variable to be predicted at time t for all $|V|$ locations. Note that $\mathbf{y}_t$ is also a column in $\mathbf{X}_{t+1}$, i.e., the values of the target variable $\mathbf{y}_t$ at the current time step become one of the predictor variables at the next time step.

In this study, we will employ a deep learning approach for the weather prediction problem. A deep learning architecture can be thought of as layers of simple functions. Each layer takes an input, $\mathbf{h}^{(l-1)}$, from the layer below and maps it to an output, $\mathbf{h}^{(l)}$, using a function $f^{(l)}$. This can be expressed mathematically as: $\mathbf{h}^{(l)} = f^{(l)}(\mathbf{h}^{(l-1)})$. We denote the dimension of the input vector to the l^{th} layer as $d^{(l-1)}$ and the dimension of the output vector of the l^{th} layer as $d^{(l)}$. The input to the first layer ($\mathbf{h}^{(0)}$) corresponds to the $\mathbf{X}_t$ whereas the output of the final layer ($\mathbf{h}^{(L)}$) corresponds to $\mathbf{y}_t$.

Our proposed WGC-LSTM framework is a novel extension of two deep learning architectures—long short-term memory (LSTM) networks and convolutional neural networks. Below we provide some background information on these architectures to motivate discussion of our proposed framework in the next section. A graphical representation of our model is provided in Figure 3.3.

3.2.1 Long Short-Term Memory Networks

One of the most common ways of capturing temporal or sequential relations with neural networks is with a variation of recurrent neural networks called a long short-term memory network (LSTM) [40]. In contrast to an ordinary recurrent neural network, LSTMs are capable of learning temporal relationships extending over long periods of time while also alleviating the vanishing gradient problem present in ordinary recurrent neural networks.

The LSTM has an internal memory in each layer, $\mathbf{c}_t^{(l)}$, that it updates at each time step. The update at time t, $\tilde{\mathbf{c}}_t^{(l)}$ depends upon the output of the layer below at that time step, $\mathbf{h}_t^{(l-1)}$ as well as the output from the same layer at the previous time step, $\mathbf{h}_{t-1}^{(l)}$. So the change of the memory is given by the equation:

$$\tilde{\mathbf{c}}_t^{(l)} = \tanh(\mathbf{W}_c^{(l)}\mathbf{h}_t^{l-1} + \mathbf{U}_c^{(l)}\mathbf{h}_{t-1}^{(l)} + b_c^{(l)}) \tag{3.1}$$

In this equation and all equations that follow, any $\mathbf{W}$, $\mathbf{U}$, or b with a subscript is a trainable model parameter.

The amount that $\tilde{\mathbf{c}}_t^{(l)}$ is allowed to modify the memory is regulated by an input gate, $\mathbf{i}_t^{(l)}$ and the amount that the memory is allowed to leak from one time step to the next is regulated by the forget gate $\mathbf{f}_t^{(l)}$. The input and forget gate are determined by the following two equations:

$$\mathbf{f}_t^{(l)} = \sigma(\mathbf{W}_f^{(l)}\mathbf{h}_t^{(l-1)} + \mathbf{U}_f^{(l)}\mathbf{h}_{t-1}^{(l)} + b_f^{(l)}) \tag{3.2}$$

$$\mathbf{i}_t^{(l)} = \sigma(\mathbf{W}_i^{(l)}\mathbf{h}_t^{(l-1)} + \mathbf{U}_i^{(l)}\mathbf{h}_{t-1}^{(l)} + b_i^{(l)}) \tag{3.3}$$

Here σ represents the sigmoid function applied component wise to ensure that each component of

the input and forget gates ranges between 0 and 1. The updated value of the memory is given by:

$$\mathbf{c}_t^{(l)} = \mathbf{i}_t^{(l)} \circ \tilde{\mathbf{c}}_t^{(l)} + \mathbf{f}_t^{(l)} \circ \mathbf{c}_{t-1}^{(l)} \tag{3.4}$$

where $\circ$ represents the Hadamard product. A portion of the memory "leaks" to form the LSTM output $\mathbf{h}_t$. The amount of memory allowed to leak is regulated by an output gate o_t, which is calculated as follows:

$$\mathbf{o}_t^{(l)} = \sigma(\mathbf{W}_o^{(l)} \mathbf{h}_t^{(l-1)} + \mathbf{U}_o^{(l)} \mathbf{h}_{t-1}^{(l)} + b_o^{(l)}) \tag{3.5}$$

$$\mathbf{h}_t^{(l)} = \mathbf{o}_t^{(l)} \circ \tanh(\mathbf{c}_t^{(l)}) \tag{3.6}$$

The final output of the model, $\hat{\mathbf{y}}_t$ is:

$$\hat{\mathbf{y}}_t = \mathbf{h}_t^{(L)} = \mathbf{w}_{prediction}^T \mathbf{h}_t^{(L-1)} + b_{prediction}, \tag{3.7}$$

where $\mathbf{w}_{prediction}$ and $b_{prediction}$ are parameters used to convert the output of the last layer of the LSTM into the final prediction.

3.2.2 Convolutional Neural Networks

Convolutional neural networks are designed to learn representations of data by taking a linear combination of each data point with its neighbors with an operation called convolution. For example, we may learn the representation of each point in a time series by considering the point itself along with the time points immediately before and after it. As an illustration, let $\mathbf{x}$ be a univariate time series of length T, $\mathbf{w}$ be a weight vector, and b is a bias. Then the t^{th} element of the convolution of $\mathbf{x}$ with $\mathbf{w}$ can be expressed as:

$$(\mathbf{x} * \mathbf{w})_t = \sum_{j=-k}^{k} \mathbf{x}_{t+j} \mathbf{w}_{j+k} + b$$

where $*$ represents the convolution operation. Note that t^{th} element of the convolution depends upon points of $\mathbf{x}$ within k steps away from the t^{th} element of $\mathbf{x}$.

In contrast, the representation learned by a fully connected neural network depends upon the entire time series rather than localized segments of it. Thus, fully connected neural networks

would require $O(T^2 \times d^{(l-1)} \times d^{(l)})$ parameters to be learned for a time series of length T, whereas convolutional neural networks with filter size k would only require $O(k \times d^{(l)} \times d^{(l-1)})$ parameters with $k << T$.

3.3 Proposed Approach

Convolution is useful to handle datasets with autocorrelated observations. For example, convolutional neural networks are often applied to images as nearby pixels are correlated with each other. Similarly, convolution has been applied to time series, where the measurements at consecutive points in time are autocorrelated. However, convolutional neural networks are typically formulated in such a way that it can be applied only to datasets where observations are taken at regular intervals. In the next section we discuss how to address this problem using graph convolution and how to incorporate the graph convolution into an LSTM.

3.3.1 Graph Convolution

Convolution can be applied to spatial data by imposing a graph structure on the data. Several different approaches to graph convolution have been discussed in the literature [51] [12] [25]. In this section we describe the approach used in this chapter. For a graph convolutional neural network with L layers, the output of each layer, $H^{(l)}$, is a $|V| \times d^{(l)}$ matrix with each row representing one of the vertices (i.e. locations) of the graph. The number of vertices in each layer will stay the same, only the dimension of the vertex representation $d^{(l)}$ changes. The graph convolution of a set of vertices, $\mathbf{H}^{(l-1)}$ with a matrix of weights, $\mathbf{W}^{(l)}$ is defined as:

$$\mathbf{H}^{(l-1)} * W^{(l)} = \mathbf{A}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)} \tag{3.8}$$

where $\mathbf{A}$ is the adjacency matrix associated with the graph. The product $\mathbf{A}\mathbf{H}_t^{(l-1)}$ will sum up the features in the neighborhood around each vertex, while right multiplying this product by $\mathbf{W}^{(l)}$ allows us to consider linear combinations of features.

We can then utilize graph convolution in the layers of a neural network. The operations

performed in each graph convolutional layer can be expressed as:

$$\begin{aligned}\mathbf{H}^{(l)} =& f^{(l)}(\mathbf{H}^{(l-1)}) \\ =& \sigma(\mathbf{H}^{(l-1)} * W^{(l)}) \\ =& \sigma(\mathbf{A}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)})\end{aligned} \tag{3.9}$$

where σ is a non-linear activation function. In our proposed approach, the adjacency matrix is shared between all layers of the network to reduce the number of model parameters.

3.3.2 Graph Convolutional LSTM

Graph convolutions allow the model to utilize spatial relationships and by incorporating them into a long short-term memory network it is possible for the model to consider spatial and temporal relationships simultaneously. We call the combination of graph convolution inside an LSTM a graph convolutional LSTM (GC-LSTM). The modified LSTM equations are:

$$\begin{aligned}\mathbf{C}_t^{(l)} &= \mathbf{I}_t \circ \tilde{\mathbf{C}}_t^{(l)} + \mathbf{F}_t^{(l)} \circ \mathbf{C}_{t-1}^{(l)} \\ \mathbf{F}_t^{(l)} &= \sigma(\mathbf{H}_t^{(l-1)} * \mathbf{W}_f^{(l)} + \mathbf{H}_{t-1}^{(l)} * \mathbf{U}_f^{(l)} + \mathbf{b}_f^{(l)}) \\ \mathbf{I}_t^{(l)} &= \sigma(\mathbf{H}_t^{(l-1)} * \mathbf{W}_i^{(l)} + \mathbf{H}_{t-1}^{(l)} * \mathbf{U}_i^{(l)} + \mathbf{b}_i^{(l)}) \\ \tilde{\mathbf{C}}_t^{(l)} &= tanh(\mathbf{H}_t^{(l-1)} * \mathbf{W}_c^{(l)} + \mathbf{H}_{t-1}^{(l)} * \mathbf{U}_c^{(l)} + \mathbf{b}_c^{(l)}) \\ \mathbf{H}_t^{(l)} &= \mathbf{O}_t^{(l)} \circ tanh(\mathbf{C}_t^{(l)}) \\ \mathbf{O}_t^{(l)} &= \sigma(\mathbf{H}_t^{(l-1)} * \mathbf{W}_o^{(l)} + \mathbf{H}_{t-1}^{(l)} * \mathbf{U}_o^{(l)} \\ &\quad + \mathbf{C}_t^{(l)} * \mathbf{V}_o^{(l)} + \mathbf{b}_o^{(l)})\end{aligned}$$

where $*$ represents graph convolution. There are two primary differences between these equations and the original LSTM formulation given in Equations 3.1-3.6. First, many of the variables are now matrices with $|V|$ rows rather than vectors. Second, all the matrix multiplications in the ordinary LSTM equations are replaced by graph convolutions.

3.3.3 Weighted Graph Convolutional LSTM

Previous work has often applied graph convolutions to datasets with known graph structure or used simple heuristics to form the graph. However, in the case of weather prediction, there are many factors that can potentially affect the spatial relationships between different locations, which makes it difficult to manually create the adjacency matrix. To overcome this problem, we propose to treat the entries of the adjacency matrix as a model parameter. This gives the model complete freedom to determine, in a data driven way, how to organize the relationships between vertices. Once this adjacency matrix is learned it can be examined for insight into weather phenomena as well as future research into climate networks.

One published work [61] has also explored the possibility of using graph convolutional LSTMs for language processing and traffic prediction applications based on diffusion convolution [25]. Specifically, the $j^{(th)}$ column of the convolution between the $|V| \times d^{(l)}$ matrix of vertex features $\mathbf{H}$ with an order 3 tensor of parameters with dimension $d^{(l-1)} \times d^{(l)} \times K$ is defined as:

$$(\mathbf{H} * \mathbf{W})_{:,j} = \sum_{i=1}^{d_{in}} g_{\mathbf{W}_{i,j}}(\mathbf{L})\mathbf{H}_{:,i},$$

where $\mathbf{L}$ is the graph Laplacian. If, $\mathbf{1}$ is the identity matrix and $T_k(\mathbf{L})$ is the order k Chebyshev polynomial of $\mathbf{L}$ and λ_{max} is the largest eigenvalue of $\mathbf{L}$ then:

$$g_{\mathbf{W}}(\mathbf{L})\mathbf{x} = \sum_{k=0}^{K-1} \mathbf{w}_k T_k(\tilde{\mathbf{L}})$$

$$\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{1}$$

However, learning the weighted graph convolution using the Chebyshev polynomial approach is a very expensive operation. This is because if we modify the graph Laplacian at each training step, then the value of λ_{max} would have to be recomputed. As a consequence, these previous approaches assume that the adjacency matrix of the graph is known and fixed unlike our proposed WGC-LSTM approach, which learns the adjacency matrix from the training data. We avoid the costly calculation of eigenvalues by using the following graph convolution operation as described in Section 3.3.1:

$$\mathbf{H} * \mathbf{W} = \mathbf{AHW}$$

where $\mathbf{W}$ is a $d^{(l-1)} \times d^{(l)}$ parameter matrix.

In our framework, the value of $\mathbf{A}$ can be trained using gradient descent based methods such as Adam [50] as long as the loss function is a differentiable function. However, the edge weights introduce an additional $|V|^2$ parameters into our model. The addition of such a large number of parameters can potentially lead to overfitting. It is worth noting, however, that as bad as the addition of $|V|^2$ model parameters may initially seem, even with a fully learned adjacency matrix this convolution-like operation still has fewer parameters than a similar fully connected network. If we are given a graph of $|V|$ vertices each with dimension $d^{(l)}$, mapping to a $d^{(l+1)}$ dimensional representation for each vertex, a fully connected neural network layer would require $|V|^2 d^{(l)} d^{(l+1)}$ parameters since every feature of each of the output vertices depends on all the features of all the vertices in the input. In contrast, using graph convolution with learned edge weights requires only $|V|^2 + d^{(l)} d^{(l+1)}$ parameters.

For the weather prediction task we expect that the vast majority of potential edges in the dataset are unnecessary as any given station will be related to just a small percentage of the other stations. One way to create sparser models is by using L1 regularization, $\lambda \|\mathbf{A}\|_1$. The choice of the value for the hyper-parameter λ determines the desired sparsity of the adjacency matrix. Another way to reduce the number of parameters is to discard some of the edges in the graph before learning. This makes sense for weather prediction, which is driven primarily by the current weather at nearby locations. Consequently, we can remove edges between distant weather stations. When we pre-sparsify the adjacency matrix by removing edges between distant stations, we call the model a **sparse WGC-LSTM**. An added bonus of using a sparse adjacency matrix is that it enables us to compute the convolution operation using sparse matrix multiplication rather than dense matrix multiplication thus reducing the amount of computation required. However, in practice deep learning models are often deployed on GPUs which cannot evaluate sparse matrix operations very quickly. To better take advantage of the unique capabilities of GPUs we propose to factorize the adjacency matrix[2] $A = U^T V$, where U and V are $k \times |V|$ dimensional matrices. Factorizing

[2]If the adjacency matrix is expected to be symmetric then this can be further simplified by factorizing A as $A = U^T U$

the adjacency matrix in this way can significantly reduce the number of model parameters while also reducing the required computation in a GPU friendly way. Computing this low rank graph convolution has time complexity $O(k \times |V| \times d + d^2)$. The low rank assumption is justified because we are dealing with spatial data and we expect the intrinsic dimensionality of the data to be much lower than the number of stations in the dataset. We call this approach **low-rank WGC-LSTM**.

3.4 Experimental Evaluation

We have performed extensive experiments using two real-world datasets to evaluate the performance of our proposed WGC-LSTM framework. In this section, we describe the datasets and baseline algorithms used along with experimental setup and the results obtained.

3.4.1 Weather Datasets

There are two real-world weather datasets are used in this study: (1) Radiosonde (weather balloon) data from the Integrated Global Radiosonde Archive (IGRA)[3] and (2) NOAA Global Surface Summary of the Day (GSOD) dataset[4]. In both datasets, we consider only measurements taken from locations in the continental United States. Furthermore, a location is included in the dataset only if less than 10% of their data is missing. Any missing values for the selected location are linearly interpolated in time. A summary of the datasets is given in Table 3.1.

For the IGRA dataset, measurements of dew point, wind speed, wind direction, temperature, and geopotential height were obtained from weather balloons released at particular heights. Our goal is to predict one of the weather variables (temperature or wind speed) twelve hours in the future based on the weather measurements from its previous time steps. The resulting dataset contains weather measurements from 67 locations, spanning the years between 1996 and 2015. Measurements from the first 10,000 time steps are used for training, the next 2,300 time steps are used as validation set, and the remaining 2,300 time steps are used as a dedicated test set. All the variables are standardized so that their mean is zero and standard deviation is one.

[3]https://www.ncdc.noaa.gov/data-access/weather-balloon/integrated-global-radiosonde-archive
[4]https://data.noaa.gov/dataset/global-surface-summary-of-the-day-gsod

Dataset	# Locations	Time Period	Training Size
IGRA	67	1996-2015	10,000
GSOD	332	1990-2016	7,800

Table 3.1: Summary of weather datasets.

For the GSOD dataset, daily measurements of minimum, maximum, and average temperature are recorded along with the maximum and average wind speed as well as average surface pressure from various weather stations. Our objective is to predict one of the weather variables (temperature or wind speed) at every selected weather station for the next day. Similar to the IGRA dataset, all variables are standardized to mean zero and unit variance. The dataset contains measurements from 332 weather stations, spanning the years from 1990 to 2016. Measurements from the first 7,800 days are used as the training set and the next 1,000 days are used as validation set. Finally, we reserve the data from another 1,000 days as test set.

3.4.2 Baseline Algorithms

We compared the performance of WGC-LSTM against the baseline methods listed below.

1. **Previous Time Step:** This baseline simply predicts the weather observations at the next time step to be the same as those at the current time step.

2. **Gaussian Process:** This baseline uses spatio-temporal Gaussian process regression. Gaussian process regression makes predictions at the next time step by taking a linear combination of the observations at previously observed time steps. We use a kernel that is the product of a spatial kernel and a temporal kernel. The spatial kernel is an RBF kernel based on the geographic distance between two stations. The temporal kernel is an RBF kernel that takes the time differences between observations as input. Only the 10 most recent observations are used to predict the next time step.

3. **Graph Convolution:** This baseline uses only $\mathbf{X}_t$ to predict $\mathbf{Y}_t$ with a graph convolutional

neural network. This approach primarily relies on spatial relationships to make predictions and has very little temporal information.

4. **Local LSTM:** This baseline trains an ordinary LSTM to predict the future weather at a given location based on weather information from its previous time steps. This is equivalent to using a graph convolutional LSTM with an identity matrix as its spatial adjacency matrix.

5. **Global LSTM** At each time step we combine the data from all weather stations into a single vector to form a multivariate time series. This multivariate time series is then used as the input to an LSTM. The LSTM will be tasked with predicting a multivariate time series of length V, where each element of the prediction vector corresponds to the forecasted weather at a particular station. In this approach, the LSTM's prediction depends not only on the weather condition at a given location, but also on the weather conditions at other locations as well. Due to the large number of parameters in the model, an L2 regularizer is placed on the LSTM parameters to avoid overfitting.

6. **GC-LSTM (Fixed Adjacency)** This corresponds to a graph convolutional LSTM approach with a fixed adjacency matrix, where the edge weights are computed based on a Gaussian kernel on the geographic distance between each pair of locations, i.e., $A_{ij} = e^{\left(-\frac{d_{ij}^2}{2\sigma^2}\right)}$.

3.4.3 Experimental Setup

All LSTMs were trained with a variant of the gradient descent algorithm called Adam [50]. Each LSTM based model has the following hyper parameters: weight of an L_2 norm regularization term, number of LSTM layers, number of hidden units. The low-rank WGC-LSTM has a hyper-parameter controlling the rank of the adjacency matrix. The rank is varied between 2 and one half of the number of stations in the dataset. Hyper-parameters were chosen through a random search [8]. The initial learning rate was set to 0.01 and periodically reduced.

	IGRA		GSOD	
Approach	Temp	Wind Speed	Temp	Wind Speed
Previous Time Step	0.8766	0.3554	0.9103	0.1739
Gaussian Process	0.8940	0.4467	0.9016	0.1888
Graph Convolution (without LSTM)	0.9094	0.5567	0.9299	0.3863
Local LSTM	0.9187	0.5400	0.9476	0.4796
Global LSTM	0.9423	0.6147	0.9523	0.5600
GC-LSTM (fixed adjacency)	0.9353	0.6117	0.9459	0.4930
WGC-LSTM (learned adjacency)	**0.9523**	**0.6412**	**0.9705**	**0.5982**

Table 3.2: Comparison of R^2 value for graph convolutional LSTM against other baseline methods.

	Gaussian Process	Graph Convolution	Local LSTM	Global LSTM	GC-LSTM	WGC-LSTM
Gaussian Process	0.0%	0.0%	0.0%	0.3%	0.0%	0.3%
Graph Convolution	100.0%	0.0%	0.6%	26.8%	6.6%	2.4%
Local LSTM	100.0%	99.4%	0.0%	44.6%	60.2%	9.6%
Global LSTM	99.7%	73.2%	55.4%	0.0%	60.2%	0.6%
GC-LSTM	100.0%	93.4%	39.8%	39.8%	0.0%	6.3%
WGC-LSTM	99.7%	97.6%	90.4%	99.4%	93.7%	0.0%

Table 3.3: Percent of stations for which the model on each row outperforms the model in the column on the temperature prediction task for the GSOD dataset.

	Gaussian Process	Graph Convolution	Local LSTM	Global LSTM	GC-LSTM	WGC-LSTM
Gaussian Process	0.0%	2.4%	0.0%	0.3%	0.3%	0.0%
Graph Convolution	97.6%	0.0%	2.1%	4.8%	13.6%	2.7%
Local LSTM	100.0%	97.9%	0.0%	13.9%	47.0%	6.9%
Global LSTM	99.7%	95.2%	86.1%	0.0%	89.2%	3.6%
GC-LSTM	99.7%	86.4%	53.0%	10.8%	0.0%	1.2%
WGC-LSTM	100.0%	97.3%	93.1%	96.4%	98.8%	0.0%

Table 3.4: Percent of stations for which the model on each row outperforms the model in the column on the wind speed prediction task for the GSOD dataset.

3.4.4 Experimental Results

The following experiments are performed to evaluate the proposed approach. First we compare the R^2 of WGC-LSTM against the baseline methods. Second, we investigate the effect of reducing the rank of the adjacency matrix. Finally, we examine the impact of varying the number of LSTM layers on the predictive performance.

3.4.4.1 Comparison against Baseline Methods

Table 3.2 shows the results of our experiments comparing graph convolutional LSTMs (WGC-LSTM and GC-LSTM) against other baseline methods. For WGC-LSTM, we use the sparse implementation of the framework and reported the results on both IGRA and GSOD datasets, with temperature and wind speed as the response variables. The results in Table 3.2 demonstrate the superiority of both GC-LSTM and WGC-LSTM on all four prediction tasks. For wind speed prediction the improvement in R^2 is greater than 3% when compared against the strongest baseline. For temperature prediction, the gain is more modest (aproximately 1%), which is not surprising as the simple baseline of using temperature from previous time step is sufficient to obtain high accuracy. The global LSTM is also capable of making accurate predictions as it considers the weather observations at multiple locations. However, the weighted graph convolutional LSTMs still outperform the global LSTM by 2% (for temperature prediction) and 4% (for wind speed prediction), which suggests that the graph convolutional LSTMs are more effective at utilizing observations from multiple locations to improve their predictions while using fewer parameters. We also observe significant performance gain using WGC-LSTM compared to GC-LSTM on all tasks, which suggests the advantages of treating edge weights as learned parameters in graph convolutional LSTM. The performance gain is less significant for temperature prediction as the R^2 is already very close to 1.

A more detailed analysis on the performance comparison is shown in Tables 3.3 and 3.4. We compare the predictive performance of the competing methods on a location by location basis. Each number represents the percentage of locations in which the method specified in the given row outperformed the method specified in the given column. We see that the proposed WGC-LSTM method is superior than all other baseline methods for the majority of the locations on the GSOD dataset. In particular, WGC-LSTM outperforms other competing methods between 90% to 99% of the locations for temperature prediction and by more than 93% of the locations for wind speed prediction.

In Figure 3.4 we plot a map of the MSE error for temperature prediction at each station. Each

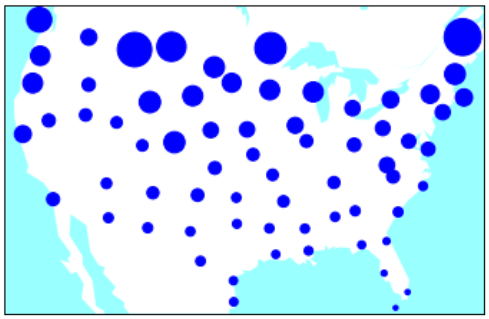


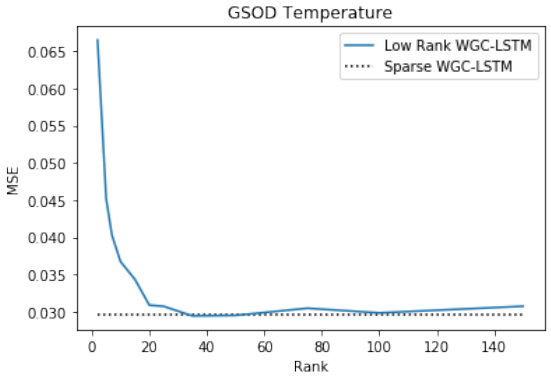
Figure 3.4: Plot of IGRA stations with station size scaled to be proportional to the cubic value of MSE of the WGC-LSTM for temperature prediction at that station. We see that there are larger errors in the north. Correlation between a station's average error and station temperature standard deviation is 0.82.

circle represents a station and the size of the circle is scaled according to the cubic value of MSE for temperature prediction at each station. Observe that the errors tend to be smaller near the coastal and southern stations and larger for those located to the north. Further analysis shows that the correlation between a station's average squared error and standard deviation of its temperature is 0.82. This suggests the lower error for stations located to the south and along the coast could be due to the less variability in their temperature compared to the northern stations.

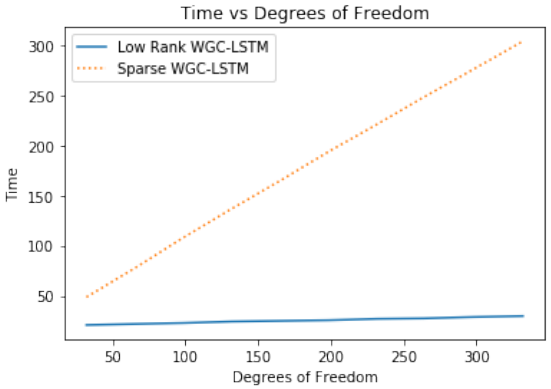
3.4.5 Low-Rank versus Sparse WGC-LSTM

This section examines the effect of reducing the rank of the adjacency matrix on predictive performance and runtime. A low rank adjacency matrix can be created by factorizing the adjacency matrix into the product of two $|V| \times k$ matrices where k is the matrix rank. By factorizing the adjacency matrix in this way we help to reduce run time while also reducing the number of model parameters. In Figure 3.5a we see that it is possible to obtain strong performance on the GSOD wind speed prediction task with a rank 40 matrix (the original matrix is 332×332).

In addition, we compare the runtime of a model with a low rank adjacency matrix against a model with a full rank adjacency matrix sparsified by removing edges between very distant locations. We see in Figure 3.5b that a low rank adjacency matrix requires much less time to compute than a sparse adjacency matrix with the same number of model parameters. This large



(a) MSE for temperature prediction as a function of adjacency matrix rank.



(b) Average time per training iteration (for the GSOD dataset) as a function of degrees of freedom. The low rank WGC-LSTM results in a much faster model than the sparse WGC-LSTM.

Figure 3.5: Comparison between low rank WGC-LSTM and sparse WGC-LSTM.

discrepency is due to the fact that the GPUs used in many deep learning tasks are not optimized for sparse-dense matrix multiplications but are optimized for dense-dense matrix multiplications.

3.4.6 Number of Layers

Next we consider the impact of the number of layers on model performance. Figure 3.6 shows predictive performance as a function of the number of LSTM layers in the network. The y-axis shows the maximal R^2 reached across all runs for graph convolutional LSTMs with a learned adjacency matrix and a given number of layers. Only 2 layers are necessary to achieve near optimal performance on the IGRA dataset while the GSOD dataset benefits from increasing the number of layers further. Since each added layer represents an additional multiplication by the adjacency matrix, adding more layers to the WGC-LSTM allows it to incorporate weather information from larger areas.

3.4.7 Learned Adjacency Matrix Visualization

In this section we visualize the learned adjacency matrix in two ways. First, we directly visualize the adjacency matrix by plotting the most important weights. Second, we use the learned adjacency

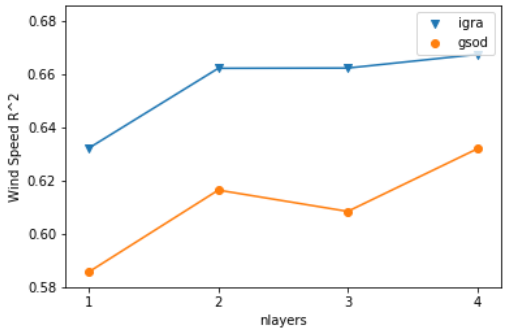


Figure 3.6: Prediction error for wind speed as a function of the number of layers in WGC-LSTM.

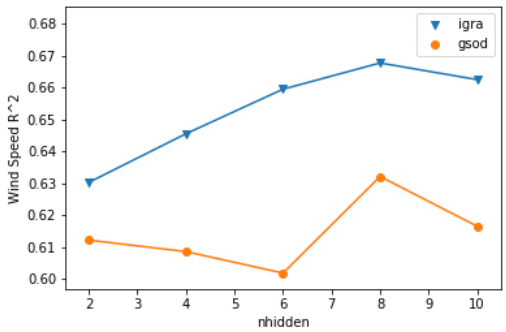
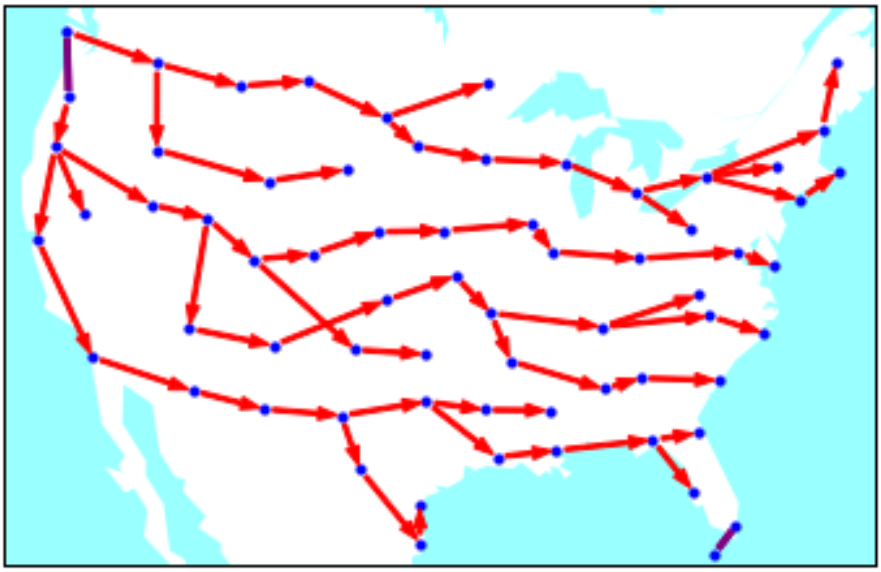


Figure 3.7: Prediction error for wind speed as a function of the number of layers in WGC-LSTM.

matrix to cluster stations. Throughout this section we compare results for models with the low rank assumption against models with a sparsity assumption.

In Figure 3.8 we visualize the edges learned by a WGC-LSTM for the IGRA temperature prediction task. To do this, we take the absolute value of the learned adjacency matrix and find the largest edge weight in each row of the adjacency matrix. The edge corresponding to each row's largest weight is then plotted along with its direction. In cases where both A_{ij} and A_{ji} would be plotted we display the edge as a purple line segment. We visualize a sparse full rank adjacency matrix that includes edges between stations less than 1400 km long and a low rank adjacency matrix with edges longer than 1400 km removed before plotting. What we observe is that the most significant edge for each station closely matches what we would expect from climate research. In particular, we see that stations are almost always most heavily influenced by stations to the west. This phenomenon is clearer for the sparse adjacency matrix than the low rank adjacency matrix.



(a) Sparse adjacency matrix

(b) Low rank adjacency matrix

Figure 3.8: Comparison between the sparse and low rank adjacency matrices learned for temperature prediction on the IGRA dataset.

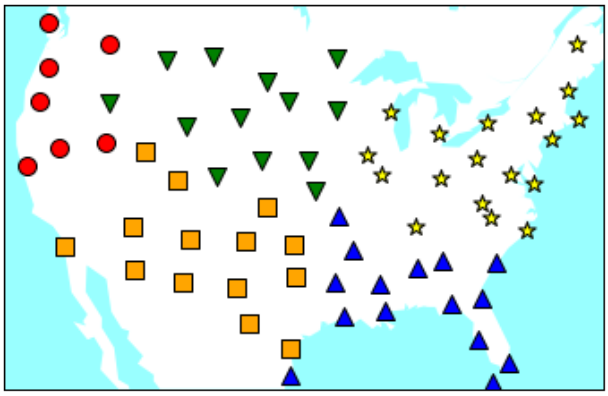
After learning the adjacency matrix it becomes possible to assign different weather stations to different regions using a clustering algorithm. Our choice of clustering algorithm depends on the structure of the adjacency matrix. For a sparse adjacency matrix, we apply spectral clustering. When using a low rank adjacency matrix, we concatenate the learned latent factors and use them as input to K-Means clustering. We compare the clusters obtained from the adjacency matrix learned in the IGRA temperature prediction task to the clusters found using a fixed adjacency matrix based on the RBF kernel computed from the geographic distance. These clusters are formed using spectral clustering. The clusters resulting from the fixed adjacency matrix are generally quite “spherical" and are spatially contiguous. The clusters derived from the sparse adjacency matrix tend to be geographically contiguous but not quite as spherical as those arising from the fixed adjacency matrix. This is most apparent when observing stations located around the Gulf of Mexico. Nevertheless, the clusters obtained from the sparse adjacency matrix are still quite similar to those found using the fixed adjacency matrix. In contrast, when the low-rank adjacency matrix is used for clustering, the stations within a cluster may be geographically distant from each other. Its possible that the clusters formed based on the low rank model are less geographically contiguous because the adjacency matrix formed by the low rank factors is a dense matrix with edges spanning the entire United States whereas the sparse adjacency matrix only has edges between nearby stations.

In both of these tasks, the sparsity assumption yielded more interpretable results. This suggests that, though low rank adjacency matrices may be much more computationally efficient, sparse adjacency matrices may still be useful in situations where the interpretability of the learned adjacency matrix is highly important.

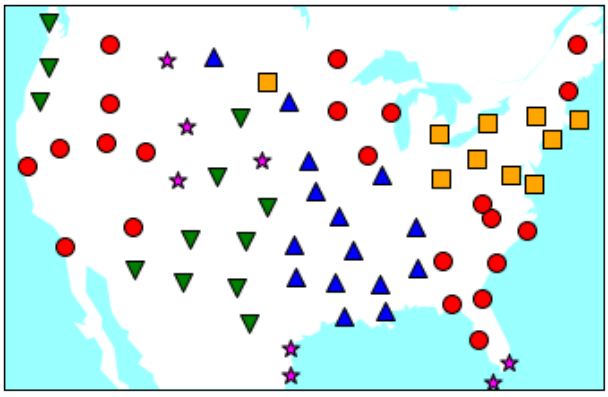
3.5 Related Work

3.5.1 Deep Learning

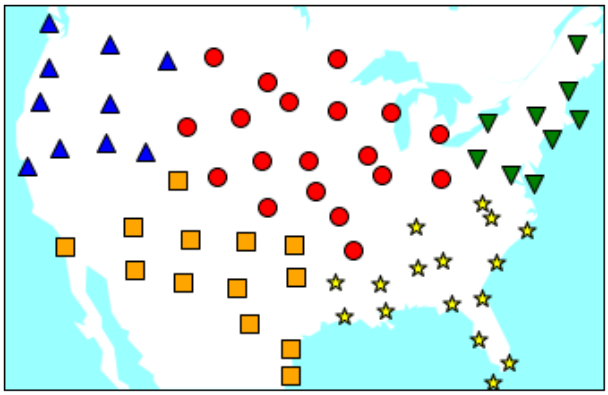
Over the past decade, deep learning has exploded in popularity. Thanks to the availability of large labeled datasets and access to cheap computing power (especially GPUs), deep neural networks



(a) Sparse adjacency matrix clustering



(b) Low rank adjacency clustering



(c) Clustering based on geographic distance

Figure 3.9: Comparing the results of clustering performed on the sparse and low rank adjacency matrices learned for temperature prediction on the IGRA dataset are used for clustering. An adjacency matrix formed from geographic distance is also clustered.

have been able to achieve strong results on challenging tasks. They have successfully been applied to object recognition [53], speech recognition [35], and machine translation [83].

To date, most work applying deep learning to spatiotemporal data has focused on sequences of gridded observations like videos. Because each sample in the sequence is gridded convolutional layers can easily be incorporated into the deep learning. For example, [80] develops that convolutional LSTM model that incorporates convolution into an LSTM and applies their model to the problem of precipitation now-casting based on sequences of gridded radar observations.

3.5.2 Graph Convolution

Many of the most successful applications of deep learning involve convolutional neural networks. Consequently, there has been a wide range of research exploring the possibility of generalizing convolutional neural networks so that they are applicable to a broader class of problems.

[12] devises a neural network architecture based on graph convolutions and apply it to graph classifications problems on synthetic datasets. Building on this, [25] use a polynomial approximation of graph convolution that is computationally efficient and localized. The effectiveness of the method is demonstrated for the classification of vertices in a graph. [51] further approximates graph convolution and also proposes a technique for applying graph convolutions to semi-supervised learning problems. Our approach to graph convolution is most similar to [51] but our approach to graph convolution is distinct from theirs in several important ways. First, [51] used a dataset with known graph structure and weights. In contrast, we learn edge weights over the course of training. Second, their approach is based on multiplication by the normalized graph Laplacian of a self-loop augmented graph and is formulated in the spectral domain. Our approach however is formulated in the vertex domain and involves multiplication by a learned adjacency matrix.

One previously published work has also used graph convolutional LSTMs for prediction. In [61] the authors use a graph convolutional LSTM for traffic prediction. We distinguish our work in the following ways. First, in [61], the authors assume that the graph structure is known. However, we show that incorporating edge weights as a part of the learning process can improve predictive

performance. In addition, [61] use a formulation of graph convolution based on random walks while the graph convolution operation employed in this chapter is formulated in the vertex domain. Since we are learning the edge weights graphs, the approach to graph convolution utilized in [61] would require computing the powers of the adjacency matrix at each training step which takes $O(|V|^3)$ time. We avoid this costly computation in our formulation.

3.5.3 Weather Prediction

The most common approach to weather prediction is based on numerical simulations [7]. In addition, data-driven approaches such as [36] [69] have also been developed. The framework in [36] contains three components: the first component consists of a regression model at each location for which they would like to produce predictions that predicts weather at that location in the future based on the historical weather measurements at that location, the second component enforces smoothness constraints using a gaussian process with a custom kernel, and the third component uses a restricted Boltzman machine to enforce physical constraints on the weather variables at each location. [80] uses a convolutional LSTM to perform precipitation now casting using ground based radar observations. In [69], the authors propose an extension of decision trees to spatiotemporal data that they evaluate on weather prediction tasks. Finally, in [54], the authors use an LSTM for soil moisture prediction.

3.6 Conclusions and Future Work

In conclusion, an approach to weather prediction using graph convolutional LSTMs was proposed. We evaluated our WGC-LSTM method against several baselines and found that it outperformed the baselines on all prediction tasks. In particular, we found that learning the edge weights gave noticeable performance improvements. We also evaluated predictive performance as a function of adjacency matrix rank and found that a relatively small rank was necessary in order to reach near optimal performance. This makes it possible to reduce the computational requirements of training and reduce the chance of overfitting without compromising prediction quality.

In this chapter we focused on making predictions using a fixed graph. Future work might explore ways to extend the framework proposed here to allow weather stations to be added or removed over time. Another possible research direction is to explore ways of leveraging datasets where different vertices have different predictor variables. For example, some vertices may contain observations of temperature and wind speed while others contain only precipitation measurements. In addition, we will also explore the possibility of designing a hybrid approach that leverages the computational benefits of the low rank structure and the interpretability yielded by sparse models simultaneously.

CHAPTER 4

CONVOLUTIONAL METHODS FOR PREDICTIVE MODELING OF GEOSPATIAL DATA

4.1 Introduction[1]

Predictive modeling of geospatial data is an important class of machine learning problem as it can be applied to a wide range of domains, from climate [80] and ecological sciences [23] to disease surveillance [39] and agricultural sciences [107]. The problem typically involves fitting a regression model to predict a future scenario (e.g., climate condition, disease incidence, or crime rate) at each location in a study region. These applications often present various modeling challenges, including the prevalence of noisy data, limited amount of labeled examples to fit accurate local models, and the need to consider the spatial relationships of the data. The limited labeled data in geospatial applications is a common challenge as the phenomenon being studied may need to be observed over long time scales to effectively predict its future behavior. Many sophisticated models that work well with access to large amounts of training data but may falter when data is limited. In addition the applications may involve complex, potentially nonlinear, spatial relationships. For example, in weather forecasting, two nearby locations are likely to experience similar weather patterns but the relationship can be altered by differences in altitude or proximity to oceans, among other reasons.

One recent technique with application to geospatial data is convolution. The strength of convolution lies in its ability to model complex non-linear relationships in the data. Convolution initially found its most widespread application in computer vision tasks such as hand-written digit recognition [60] but more recently has been applied to a wider range of domains, including climate downscaling [94] and precipitation nowcasting [80]. However, current methods are typically used to model spatial relationships that exist among predictor variables, such as the pixels of an image.

[1]This chapter is adapted from a previously published paper: Tyler Wilson, Pang-Ning Tan, and Lifeng Luo, "Convolutional Methods for Predictive Modeling of Geospatial Data". In Proceedings of the SIAM International Conference on Data Mining (SDM-2020), Cincinnati, OH (2020).

We termed this approach as *convolution on input* or $Conv_x$.

The recent success in applying multi-task learning to geo-spatial prediction problems has demonstrated the utility of modeling the relationship between the parameters of different local models [64]. Specifically, when applying multi-task learning to geo-spatial prediction, the different tasks (i.e., predictions at different locations) are often represented as vertices of a graph while the relationship between tasks is represented by the edges of the graph. In this chapter, we propose an extension of convolution to model the relationships between the local model parameters of the different tasks. This is accomplished by using a graph convolutional neural network to map from a low dimensional embedding of each location to the location's model parameters. We refer to this approach as *convolution on parameters* or $Conv_w$.

Our use of convolution raises several interesting research questions: When should we perform convolution on predictors? When should we use convolution to capture relationships between model parameters? Are there cases where convolution can be effectively used to represent relationships among both predictors and model parameters? These issues will be explored in more details in the remainder of this chapter.

To address these issues, we propose a flexible deep learning framework for geospatial prediction. The framework assumes each location has its own local model parameters, which can be derived using a graph convolutional neural network, while simultaneously learning a low dimensional representation for each location. This convolution on parameter ($Conv_w$) approach can thus be viewed as an implementation of traditional multi-task learning [14] using graph convolutional neural network. More importantly, the framework can also accommodate for convolution on input, $Conv_x$, or a hybrid of both, $Conv_{xw}$. We also provide guidance on how to determine when each type of convolution should be used in geospatial prediction problems and demonstrate the effectiveness of our framework on both synthetic and real-world datasets.

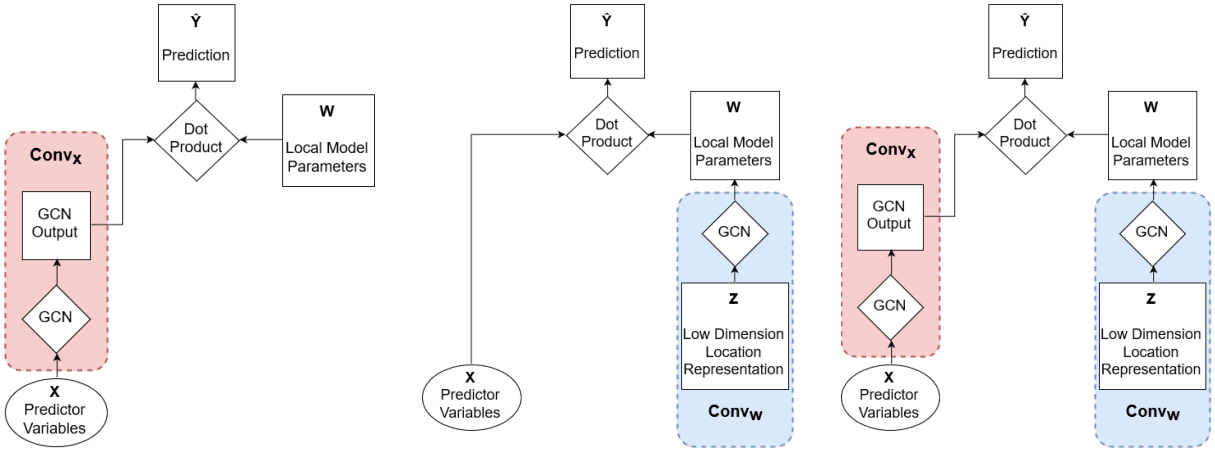
4.2 Related Work

Geospatial prediction encompasses a wide range of important problems including weather forecasting [36], disease spread detection [39], and crop yield prediction [107]. Such prediction often involves the creation of a separate local model at each location within the study domain as a single global model might not effectively capture the spatial heterogeneity of the data. Conversely, training an accurate local model can be challenging when there are limited training examples available at each location. In this case, it would be better to jointly model the prediction tasks for all locations using multi-task learning (MTL) techniques [14]. MTL attempts to improve generalization errors by exploiting the relationships between different tasks. The approach has been successfully applied to many applications, such as climate [64] and soil moisture prediction [105].

Convolutional neural networks (CNNs) have proven to be a powerful technique for modeling spatial relationships in data. The idea was first applied to computer vision problems [53, 60], where it was used to model the relationships between pixels. However, CNN is also capable of capturing spatial relationships in other applications including precipitation nowcasting [80] and climate downscaling [94]. One major drawback of CNNs is that they are only capable of handling spatial relationships in data with a grid-like structure. This assumption was relaxed in work that generalized convolution to graphs. Bergstra and Bengio [8] proposed spectral graph convolutional neural networks, which are extended by Defferrard et al. [25] with localized filters. Kipf and Welling [51] proposed an approximation of graph spectral convolution. Graph convolutional methods have since been applied to spatial problems such as weather prediction [101].

4.3 Convolutional Methods for Modeling Geospatial Data

Let $\{(\mathbf{X}_l, \mathbf{y}_l)\}_{l=1}^{|L|}$ be the dataset collected for a set of geo-referenced locations L, where $\mathbf{X}_l \in \mathbb{R}^{T_l \times d}$ denotes the values of the input (predictor) variables and $\mathbf{y}_l \in \mathbb{R}^{T_l}$ denotes the values of the output (target) variable at location l for T_l time steps. Let $\mathbf{A} \in \mathbb{R}^{|L| \times |L|}$ be the adjacency matrix for modeling spatial dependencies between different locations. Our goal is to learn a target function $f_l : \mathbf{X}_l \rightarrow \mathbf{y}_l$ for each location. Each f_l is characterized by its local regression parameter, $\mathbf{w}_l \in R^d$, which can



(a) $conv_X$ (b) $conv_W$ (c) $conv_{XW}$

Figure 4.1: Three different architectures for performing convolution on geospatial data.

be concatenated column-wise to form a weight matrix $\mathbf{W} \in R^{d \times |L|}$. This section presents three different approaches for applying convolutional methods to learn $\mathbf{W}$, as shown in Figure 4.1.

4.3.1 Convolution on Input ($Conv_x$)

The use of convolutional neural networks (CNNs) was pioneered in computer vision applications to model relationships among pixels in images. Specifically, small filters are applied across the pixels of the input image so that, at each layer of the convolutional neural network, each pixel of the response map depends upon a small region of pixels from the layer below. By stacking multiple such layers on top of one another, each pixel in the final layer of the convolutional neural network will depend upon a number of pixels surrounding it and as a result the learned function captures the inter-dependencies among the pixels in the input image.

The application of CNN to geo-spatial data can be motivated in a similar way. By feeding the input variables from multiple locations through the network layers, the resulting function will capture the spatial relationships among the predictor variables at different locations. We thus refer to this approach as convolution on input. One major drawback of traditional CNNs is that they require the input to possess a gridded structure, like the pixels of an image. Thus, they may not be appropriate for datasets in which the locations of the observations can be arbitrary and do not

conform to a regular grid structure. Graph convolutional networks (GCNs) are a generalization of traditional CNNs, which allows the data to have a graph structure. In the context of geo-spatial prediction, each vertex of the graph corresponds to one location and each edge represents the relationship between a pair of locations.

The overall structure of a model that performs convolution on the predictors is show in Figure 4.1a. The predictors at all locations are fed through a GCN, which computes a high level representation of the predictors at each location (denoted as GCN output in Figure 4.1a). The learned representation at each location is thus dependant upon its own predictor values as well as those of its neighbors. Assuming a linear model, we can compute a dot product between the GCN output at each location with its local regression parameter $\mathbf{w}_l$ to generate a prediction vector $\hat{\mathbf{y}}_\mathbf{l}$ at each location.

We employ a Res-Net like GCN architecture [38] to map the input predictors to its output representation. Specifically, the i-th layer of GCN hs the following form:

$$\mathbf{H}_i = \mathbf{A}\sigma\big(\mathbf{A}\mathbf{H}_{i-1}\mathbf{U}_i\big)\mathbf{V}_i + \mathbf{H}_{i-1} \tag{4.1}$$

where $\mathbf{H}_i$ is the output of the i-th layer, $\mathbf{A}$ is the adjacency matrix, $\mathbf{U}_i$ and $\mathbf{V}_i$ are matrices of the GCN weight parameters, and $\sigma(\cdot)$ is an activation function. The right multiplication by $\mathbf{U}_i$ and $\mathbf{V}_i$ compute a linear transformation of the features at each location while the left multiplication by $\mathbf{A}$ allows the model to consider the features at neighboring locations. Note that $\mathbf{U}_i$ and $\mathbf{V}_i$ in each layer are the model parameters to be estimated during training while $\mathbf{A}$ is assumed to be known.

The adjacency matrix is computed based on the geographic distance between every pair of locations:

$$\mathbf{A}_{i,j} = exp\left[-\frac{dist(l_i, l_j)}{k}\right] \tag{4.2}$$

where $dist(\cdot,\cdot)$ is the Haversine distance and k is a hyperparameter.

Let $\mathbf{g}(\mathbf{X};\mathbf{A})_l \in \mathbb{R}^{T_l \times D}$ denote the GCN output associated with location l and $\mathbf{w}_l \in \mathbf{R}^D$ is its corresponding local regression parameter. The $Conv_x$ framework is trained to minimize the

following loss function:

$$\sum_{l=1}^{L} \|\Phi_l(\mathbf{X})\mathbf{w}_l - \mathbf{y}_l\|^2, \quad \text{s.t. } \Phi_l(\mathbf{X}) = \mathbf{g}(\mathbf{X}; \mathbf{A})_l \tag{4.3}$$

The parameters to be estimated are the weights V_i's from all layers of the GCN as well as the local regression parameter w_l's. All parameters are trained simultaneously in an end-to-end fashion to minimize the loss.

4.3.2 Convolution on Parameter ($Conv_w$)

In the previous subsection, we described how convolution can be used to model the spatial relationships among predictor variables. However, spatial relationships may exist between parameters of the local models. One way of enforcing the spatial relationship between models is by incorporating a graph Laplacian regularization term [104] into the loss function of the learning algorithm:

$$\sum_{l=1}^{L} \|f_l(\mathbf{X}_l; \mathbf{w}_l) - \mathbf{y}_l\|^2 + \lambda \sum_{i,j=1}^{L} \mathbf{A}_{i,j} \|\mathbf{w}_i - \mathbf{w}_j\|^2 \tag{4.4}$$

The graph Laplacian regularization ensures that the local model parameters vary smoothly over space and that any sharp discontinuities between the parameters at two nearby locations will be heavily penalized.

Alternatively, we can apply graph convolution to model the spatial relationship between the parameters, as shown in Figure 4.1b. Specifically, each location is represented by a low dimensional vector, $\mathbf{z}_l \in R^m$, where m is a hyperparameter representing the dimensionality of the embedding vector. We can then learn a function mapping from this location embedding vector $\mathbf{z}_l$ to the local regression parameters, $\mathbf{w}_l$, which we refer to as "**convolved linear weights**." By using a GCN, the output regression parameter for each location will also depend on the location embedding of its nearby neighbors. This makes it possible to pool information from nearby models as a way of combating overfitting when training samples are limited.

Our GCN is trained to minimize the following loss:

$$\sum_{l=1}^{L} \|\mathbf{X}_l\mathbf{w}_l - \mathbf{y}_l\|^2 \quad \text{s.t. } \mathbf{w}_l = \mathbf{g}(\mathbf{Z}; \mathbf{A})_l \tag{4.5}$$

By using a non-linear function to map the location embedding to its regression parameters, this gives it more flexibility in modeling relationship between regression parameters compared to graph Laplacian regularizer.

$Conv_w$ differs from $Conv_x$ in two important ways. First, in $Conv_w$, the convolved linear weights are mapped from the location embeddings using a GCN whereas the local linear regression weights in $Conv_x$ are actually model parameters learned over the course of training using Eq. (4.3). Second, the predictors in $Conv_x$ are spatially transformed using a GCN before being multiplied to each location's local regression weights. In contrast, $Conv_w$ does not perform any spatial transformation on its predictors before multiplying them to the convolved linear regression weights.

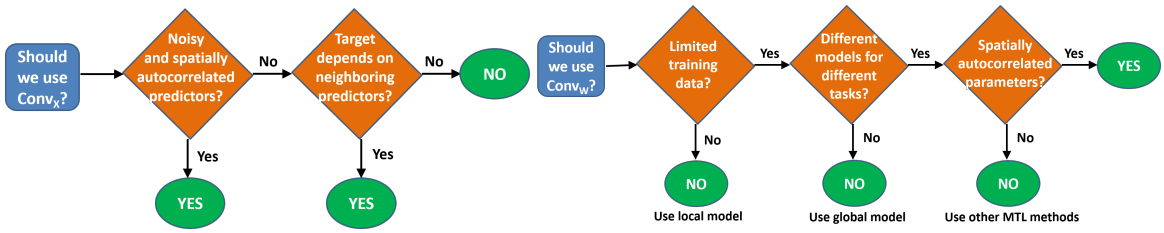
During each forward pass, $Conv_w$ will calculate the new convolved linear regression weights, $\mathbf{w}_l = \mathbf{g}(\mathbf{Z}; \mathbf{A})_l$ based upon the current value of $\mathbf{Z}$. These weights are used to predict the value of $\mathbf{y}_l$ at each location. The mean squared error of these predictions is then averaged across all observations and all stations. Since $\mathbf{w}_l$ is actually the output of a neural network, it will be updated during each backward pass via gradient descent. The GCN parameters and location embeddings are trained simultaneously in an end-to-end fashion to minimize the mean squared error of the predictions.

4.3.3 Hybrid Convolution Approach ($Conv_{xw}$)

In addition to performing convolution on $\mathbf{X}$ alone or $\mathbf{W}$ alone, it is possible to apply convolution on both X and W which allows us to leverage the advantages of both methods. In this case the loss function for $Conv_{xw}$ is:

$$\begin{aligned} \min \quad & \sum_{l=1}^{L} \|\Phi_l(\mathbf{X})\mathbf{w}_l - \mathbf{y}_l\|^2, \\ \text{s.t.} \quad & \mathbf{g_1}(\mathbf{X}_l, \mathbf{A}),\ \mathbf{w}_l = \mathbf{g_2}(\mathbf{Z}_l, \mathbf{A}) \end{aligned} \tag{4.6}$$

Estimating $\mathbf{y}$ in this way requires two separate graph convolutional neural networks, $\mathbf{g}_1$ and $\mathbf{g}_2$, one for the predictors, and the other for computing the convolved linear weights. In our implementation we assume that both GCNs share the same adjacency matrix. The parameters to be learned over



(a) $Conv_x$ (b) $Conv_w$

Figure 4.2: Flow charts showing when different kinds of convolution are applicable

the course of training are the weights internal to the two GCNs as well as the location embeddings, $\mathbf{Z}$. All parameters are trained simultaneously in an end-to-end fashion.

4.4 Choice of Convolutional Method

Below, we discuss scenarios in which the different kinds of convolutional methods are expected to be useful. We argue that the utility of convolutional methods often depends on the following 3 criteria: (1) size of local training sets, (2) degree of spatial autocorrelation, and (3) noise level of the data. While these are by no means the only possible criteria, they are generally applicable to many geospatial applications. Our suggestions on choosing the right convolutional methods to use are summarized in the flow chart shown in Figure 4.2.

4.4.1 When $Conv_x$ is useful?

$Conv_x$ is designed to capture relationships among the predictor variables. Specifically, each layer of convolution incorporates information not just from a single location, but from nearby locations as well. By stacking multiple convolutional layers on top of one another, the architecture is able to consider relationships between predictors in progressively larger regions. $Conv_x$ can thus be used to capture broad-scale spatial patterns in geospatial data. For example, in weather forecasting, the broad-scale patterns may provide information about the high or low pressure regions and their frontal zones—information that are useful to determine the local weather conditions. However, even in situations where the output at a given location depends only on the predictors at that location it may still be useful to apply $Conv_x$. For example, if the values of the predictor variables in X

are noisy, then performing convolution on the input may allow us to denoise the data and train more accurate prediction models at each location so long as the values of the predictor variables are spatially auto-correlated. $Conv_X$ will generally perform better with more training data even if that data is split across many different locations. Figure 4.2a shows a flowchart for determining whether $Conv_x$ should be applied to a geospatial prediction problem.

4.4.2 When $Conv_w$ is useful?

Unlike $Conv_x$, $Conv_w$ can be thought of as a form of multi-task learning as it enables us to capture relationships between model parameters at different locations. Thus, it is best applied in those situations where traditional multi-task learning typically excels. This suggests that $Conv_w$ is best utilized when (1) the prediction tasks are distinct enough that a single global model would perform poorly and (2) there are insufficient samples to train the local model at each location accurately. $Conv_w$ also makes the additional assumption that there exists spatial relationships between the model parameters at different locations. A flowchart for determining whether $Conv_w$ should be applied is shown in Figure 4.2b.

4.4.3 When $Conv_{xw}$ is useful?

In some situations we may wish to perform convolution on both the input and model parameters. This will most frequently be the case when each location should be associated with its own predictive function but that function should be based on relationships among predictors at different locations. The main drawback of combining the two forms of convolution is the additional model parameters. Convolution on parameters is useful primarily in situations with limited training data, in which case the additional model parameters introduced by the convolution on input makes it more susceptible to overfitting. However, as the parameters introduced by convolution on input are also shared between tasks, this may help regularize the parameters especially if there are sufficient number of autocorrelated tasks (locations).

Models	DS1	DS2	DS3
$Conv_w$	**0.832**	0.167	0.083
$Conv_x$	0.8094	**0.891**	0.491
$Conv_{xw}$	0.559	0.809	**0.524**

Table 4.1: R-squared results for synthetic data.

4.5 Experimental Evaluation

We have conducted extensive experiments using both synthetic and real-world data to demonstrate the effectiveness of convolutional methods.

4.5.1 Experiments on Synthetic Data

To test the hypotheses suggested by the flowchart shown in Figure 4.2, we have created three synthetic datasets, each of which is intended to demonstrate a situation where $Conv_w$, $Conv_x$, or $Conv_{xw}$ is useful. All datasets have a fixed test set of size 250 examples. Below, we describe how each dataset is created.

1. **DS1 (Correlated Task Model Parameters)** This dataset is created to demonstrate the effectiveness of $Conv_w$. Specifically, it has a small training set size (8 examples per location) and the model parameters are spatially autocorrelated. We first create a graph on a 30 by 30 grid to represent 900 locations. For each vertex, we generate a set of 8 40 dimensional vectors to represent the predictor variables, where each component of the vector is drawn from a standard normal distribution. We then generate the regression weights, **w**, using a Gaussian process with a covariance matrix determined by an RBF kernel of the Euclidean distance between locations and a mean value that increases linearly moving from left to right. Since there are 40 predictors, we must generate 40 separate samples for each location. To ensure that the regression weights are significantly different, we apply a separate scaling factor, v_d, drawn from the standard normal distribution, to each feature coefficient and apply the scaling factor uniformly to the w_d for all locations. Finally, the ground truth values of the target variable are generated by applying a dot product between the predictors and regression

	IGRA	GSOD	GHCN
# of Stations	67	332	13,529
Time Range	1996-2015	1990-2016	1970-2015
Frequency	12 hourly	daily	monthly
#Samples/Station	14,610	9,862	551

Table 4.2: Summary of real-world datasets.

	GHCN			IGRA		GSOD		
	TMin	**TMax**	**Prcp**	**Wspd**	**Temp**	**MaxTemp**	**MaxSpd**	**Wspd**
Local Linear	0.597	0.602	0.269	0.269	0.628	0.474	0.024	-0.014
Global Linear	0.585	0.554	0.297	0.312	0.655	0.571	0.118	0.130
MALSAR	0.613	0.622	0.303	0.311	0.658	0.567	0.118	0.152
$Conv_w$	0.611	0.617	0.304	0.315	0.660	0.576	0.123	0.136
$Conv_x$	0.619	0.632	0.295	0.355	**0.717**	0.575	0.074	0.055
$Conv_{xw}$	**0.620**	**0.649**	**0.313**	**0.362**	0.710	**0.603**	**0.126**	**0.156**

Table 4.3: Comparison of convolutional methods against other baselines using R-squared as evaluation metric.

weights. The predictor and target values are both standardized per location and noise is added to observations of y.

2. **DS2 (Correlated Predictors)** We create a graph on a 10 by 10 grid. Similar to DS1, the predictor values are drawn from the same Gaussian process except their dimensionalities are equal to 5 (instead of 40) and noise are added to the features. The training set size is 250 samples per vertex. As the predictors are noisy and their ground truth values are correlated, we expect $Conv_x$ will be most effective on this dataset.

3. **DS3 (Correlated Task Parameters and Predictors)** This dataset is created using a combination of the techniques used in DS1 and DS2. The weights, w are generated from a Gaussian process as in DS1 and the predictors are generated from a Gaussian process as in DS2. Noise are added to both the predictors and response values. As in DS1, the training set size is limited. $Conv_{xw}$ is expected to perform well on this dataset.

4.5.1.1 Experimental Results on Synthetic Data

We reported the performance of all three convolutional methods on the synthetic datasets in Table 4.1. We use R-squared as our evaluation measure. R-squared is a widely-used statistical measure to determine the proportion of variance in the target variable explained by the predictor variables. The results agree with our previous expectation. First, $Conv_w$ outperforms other methods on DS1 because of the limited training data and the spatial autocorrelations among the regression weights introduced by the Gaussian process. $Conv_x$ achieves the best performance on DS2 due to the correlations and noisiness of the predictor variables while $Conv_{xw}$, as expected, performs best on DS3, which had limited training samples, correlated and noisy predictors, as well as correlated regression weights. Together, these results validate our hypothesis in Section 4.4 regarding when to use which form of convolution.

4.5.2 Experiments on Real-world Data

In this section we examine the predictive performance of the different forms of convolution on the following three real world datasets (details are in Table 4.2):

1. **GHCN**: This is a monthly gridded dataset, where the predictors are coarse-scale variables obtained from the NCEP re-analysis project (e.g., convective precipitation rate, solar radiation flux, relative humidity, and sea level pressure) while the target variable corresponds to fine scale observations of temperature and precipitation from the Global Historical Climatology Network (GHCN)[2].

2. **IGRA**: This is a weather dataset from the Integrated Global Radiosonde Archive (IGRA)[3]. The task here is to predict the weather at each station in the next time step given its past conditions over the previous three time steps. The predictors used include dew point, wind

[2]https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets/global-historical-climatology-network-ghcn

[3]https://www.ncdc.noaa.gov/data-access/weather-balloon/integrated-global-radiosonde-archive

speed, temperature, and geopotential height with temperature and wind speed also serving as target variables.

3. **GSOD**: This is another weather dataset from NOAA Global Summary of the Day website[4]. The task here is similar to IGRA. The predictors include minimum and maximum temperature, maximum wind speed, and mean sea level pressure whereas the target variables are maximum wind speed, average wind speed and maximum temperature.

4.5.2.1 Experimental Setup

We compare the three variations of convolutional methods ($Conv_X$, $Conv_W$, and $Conv_{XW}$) against the following baseline methods:

1. **Global Linear Model**. This corresponds to fitting a single ridge regression model on the combined data from all locations.

2. **Local Linear Model** - This corresponds to training an independent ridge regression model at each location. The hyper-parameters for each location are tuned independently.

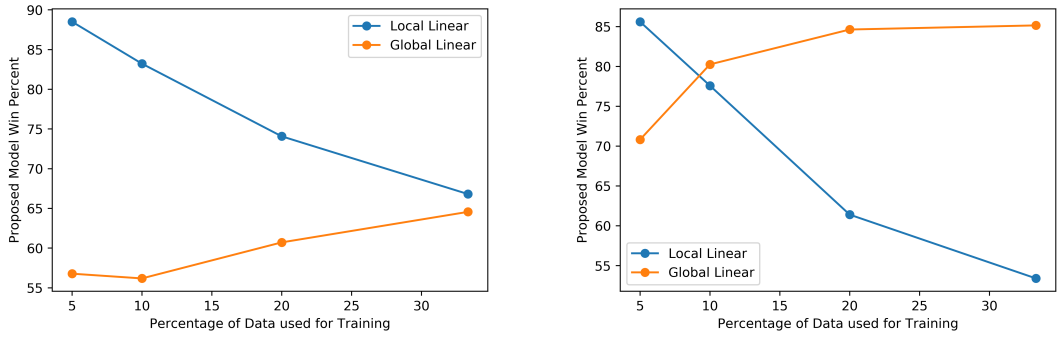
3. **SRMTL** - This is the MALSAR [115] implementation of sparse graph Laplacian regularized multi-task learning.

For all baseline and convolutional methods, we perform a random search over the hyper-parameters [8]. We also fixed both the validation and test set sizes to be 33% each for all 3 datasets. As the performance of the models depend on the amount training data, we vary the training set size from 5% up to 33%.

4.5.2.2 Experimental Results

In Table 4.3 we compare the predictive performance of the different forms of convolution against our baselines. Results are reported using 20% training data (for GHCN) and 1% training data (for

[4]https://catalog.data.gov/dataset/global-surface-summary-of-the-day-gsod



(a) Precipitation

(b) Maximum Temperature

Figure 4.3: Proportion of stations in which $Conv_w$ beats the baseline methods as the size of the training set varies.

IGRA and GSOD). We use a higher percentage for GHCN since it is monthly data with fewer number of observations per location. We find that the best performing model across seven of the eight prediction tasks is $Conv_{xw}$. The strong performance of $Conv_{xw}$ can be attributed to its ability to leverage the strengths of both convolution on input and convolution on parameters. Furthermore, $Conv_x$ consistently underperforms compared to $Conv_{xw}$, which suggests its tendency to overfit when there is limited training data. This problem can be alleviated by $Conv_{xw}$ with its convolution on parameter. The strong performance of $Conv_{xw}$ in comparison to $Conv_w$ can be attributed to its ability to incorporate predictors from multiple locations when making predictions via its convolution on input.

Of the four methods that use only local predictors at a single location to make predictions (local linear, global linear, SRMTL, and $Conv_w$), we see that either SRMTL or $Conv_w$ always performs the best depending on the dataset. The performance of SRMTL and $Conv_w$ are comparable to each other, which is not surprising as both are MTL methods. Figure 4.3 further breaks down the performance of the local linear, global linear, and $Conv_w$ models on a station by station basis for precipitation and temperature prediction on the GHCN dataset. The results suggest that when the amount of data is small (around 5%), $Conv_w$ outperforms the local model over 80% of the time while outperforming the global model at more than 55% of the stations. As training data increases,

	TMin	TMax	Prcp
Linear $Conv_x$	0.311	**0.641**	0.619
Non-Linear $Conv_x$	0.304	0.630	**0.631**
Linear $Conv_{xw}$	**0.322**	0.633	0.618
Non-Linear $Conv_{xw}$	0.321	0.639	0.630

Table 4.4: Comparing the R-squared values of linear and non-linear convolution on inputs for GHCN data.

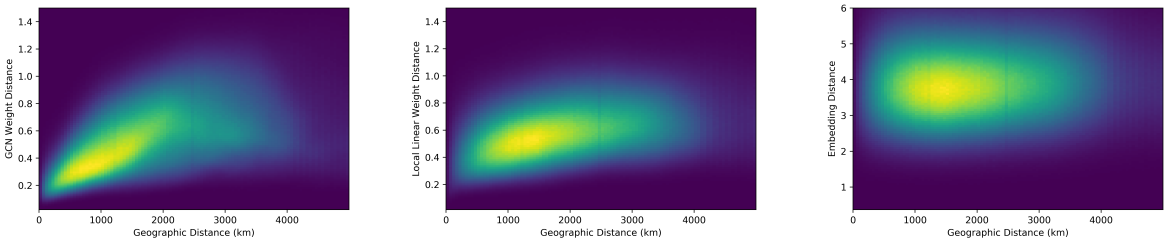
$Conv_w$ outperforms the global model at an increasing proportion of locations while still faring well against the local model baseline.

4.5.2.3 Ablation Study

Throughout most experiments, we employed GCNs with non-linear activation functions. However, in the case of $Conv_x$ and $Conv_{xw}$ this non-linear activation function in the convolution applied to inputs will make the final prediction a non-linear function of the inputs. To demonstrate that the performance gain from applying convolution to inputs is due to the utilization of additional spatial information and not primarily because the predictions are a non-linear function of the inputs, we compare the predictive performance of convolution on inputs with linear and non-linear activation functions. Results on the GHCN tasks with 33% of data used as training are shown in Table 4.4. We find that the difference in R-squared between the best performing linear and non-linear model differ by at most 0.02, which indicates that the benefits provided by performing convolution on inputs do not depend solely on non-linearity of the convolution.

4.5.2.4 Embedding Visualization

$Conv_w$ learns a separate embedding vector for each location. At first glance, one may expect that the location embedding vectors $\mathbf{z}_l$ should exhibit spatial autocorrelation. Alternatively, it may be the case that spatial auto-correlation does not exist in the location embeddings but instead results from the repeated application of graph convolution, in which case we would expect to find spatial auto-correlation in the convolved linear regression weights $\mathbf{w}_l$ instead. To verify this, in Figure 4.4,



(a) Convolved weights of $Conv_w$ and geographic distance (b) Local linear regression weights and geographic distance (c) Location embeddings of $Conv_w$ and geographic distance

Figure 4.4: Relationship between location embeddings, regression weights, and geographic distance.

	TMin	TMax	Prcp
Convolved Weights	0.277	0.431	0.082
Local Linear Weights	0.147	0.322	0.090
Local Embeddings	0.038	0.059	-0.001

Table 4.5: Correlation between regression weights and local embeddings with geographic distance.

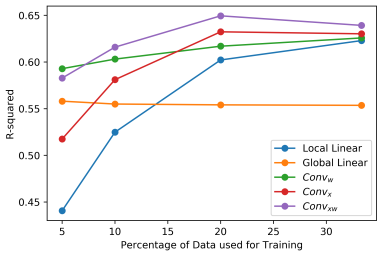
we plot the 2-d heatmaps that show the relationship between pairwise geographic distance and two other quantities: Euclidean distance of the location embeddings and Euclidean distance of their convolved linear weights, for the maximum temperature GHCN data. We plot this relationship only for pairs of stations whose geographic distance is less than 5000 kilometers because beyond this threshold the relationship between geographic distance and the other distances is reduced.

Figure 4.4c shows that there is little relationship between the location embedding distance and geographic distance. In contrast, the convolved linear weights $\mathbf{w}_l$ derived by $Conv_w$ have a much stronger relationship with geographic distance (see Figure 4.4a). This can be further verified by computing the Pearson correlation between each of these values and geographic distance as shown in Table 4.5. We see that models trained to predict maximum or minimum temperature exhibit a stronger relationship between geographic distance and the convolved linear weights than they do with local linear regression weights or location embeddings. The result suggests that the location embeddings themselves do not capture the spatial autocorrelation; instead, the GCN is responsible for deriving spatialy autocorrelated convolved linear weights from uncorrelated embeddings.

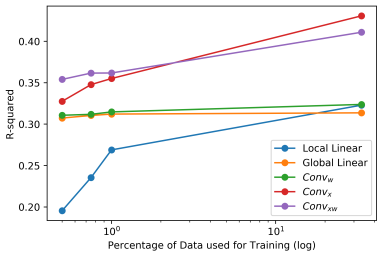
4.5.2.5 Discussion

By comparing the performance of the different forms of convolution across these three datasets we can validate our advice about which scenarios favor which form of convolution. To this end, we visualize the performance of the predictive models when different amounts of training data is available in Figure 4.5. Figure 4.5a shows that when there is an abundance of training data, a local model outperforms the global model on the maximum temperature prediction task, but when training data is limited, the local model is outperformed by the global model. This suggests that a more data efficient model may be able to outperform both the global and local linear models when training data is limited. Since the local linear regression parameters are spatially autocorrelated (see Figure 4.4b), convolution on model parameters is a natural way to improve predictive performance when data is limited. We also see that when there is sufficient data, $Conv_x$ achieves strong performance on the GHCN maximum temperature prediction task. This is not surprising since climate data tends to be noisy and spatially auto-correlated. Combining convolution on inputs and convolution on parameters improves the data efficiency of $Conv_x$ so that $Conv_{xw}$ outperforms $Conv_x$ especially when data is limited.

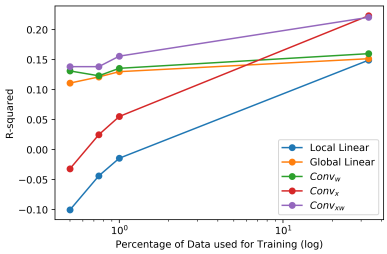
In Figure 4.5b and Figure 4.5c we show how the predictive models perform with varying amounts of training data on the IGRA and GSOD wind speed prediction tasks respectively. Unlike GHCN, the performance difference between a local linear model and a global linear model is quite small even when training data is around 33%. This suggests that it will be difficult for the local model to outperform the global model when training data is limited ($\ll 33\%$). This explains why $Conv_w$ performs so similarly to the local linear and global linear models on the IGRA and GSOD prediction tasks. The performance improvements achieved by convolution on inputs on the IGRA and GSOD datasets is even greater than the performance improvements provided by convolution on inputs on the GHCN tasks, which is likely due to future weather being driven by large-scale weather patterns in addition to small-scale patterns.



(a) GHCN Max Temperature



(b) IGRA Wind Speed



(c) GSOD Wind Speed

Figure 4.5: Performance comparison between convolutional and baseline methods on the real-world datasets.

4.6 Conclusion

In this chapter we explored the application of convolutional methods to geospatial prediction problems. We distinguished between the conventional practice of using convolution to model relationships among predictor variables and our proposed technique for using convolution to model relationships between model parameters. We provide guidance on situations when different varieties of convolution can be effectively applied and validate them on several synthetic and real-world datasets.

CHAPTER 5

A DEEP LEARNING APPROACH FOR MODELING GEOSPATIO-TEMPORAL EXTREME EVENTS

5.1 Introduction[1]

Extreme geospatio-temporal events such as flooding, heat waves, and droughts are destructive natural forces with the potential to cause devastating losses in property and human lives. According to NOAA's National Center for Environmental Information, as of July 2021, there were 8 billion dollar weather/climate disaster events in 2021 alone, incurring close to $30 billion in total losses. Given the severity of their impact, accurate modeling of extreme events are therefore critical to provide timely information to the public threatened by such hazards and to minimize the risk for human casualties and property destruction.

Numerous methods have been developed in the past for modeling extremes. This includes outlier detection methods [10, 17, 18, 19], where the goal is not to predict future extreme events but to detect them retrospectively from observation data after they have occurred. Statistical approaches based on extreme value theory [48, 49, 67] are also commonly used to infer the statistical distribution of the extreme values. Another approach is to cast the prediction of extreme events as a supervised learning problem [56, 71], which is the approach used in this chapter. Specifically, we are interested to predict the conditional distribution of excesses over a threshold (e.g., monthly precipitation or temperature that exceeds their 95th percentile) at various spatial locations. However, predicting the conditional distribution of such excesses is a challenging problem due to their rare frequency of occurrence. In addition, the predictive model must consider the complex spatial relationships between events at multiple locations. Identifying such complex and potentially nonlinear interactions among the predictors is a challenge that must be addressed.

[1]**This chapter is adapted from a previously published paper: Tyler Wilson, Pang-Ning Tan, Lifeng Luo, Asadullah Galib. "DeepGPD: A Deep Learning Approach for Modeling Geospatio-Temporal Extreme Events". To appear in Proceedings of 36th AAAI Conference on Artificial Intelligence (AAAI-2022), Vancouver, Canada (2022).**

In recent years, there have been growing interests in developing deep learning algorithms to address various spatio-temporal modeling problems [66, 80, 101]. For spatial data, one emerging technique that can effectively handle spatial relationships in the data is convolutional neural network (CNN). CNN initially found its application in computer vision tasks, but has since been successfully applied to a wider range of problems, including climate downscaling [94], precipitation nowcasting [80] and hail prediction [29]. However, despite their growing body of literature, there has been scant research on spatio-temporal deep learning for modeling extreme events.

Non-parametric deep learning methods are generally ineffective to infer the distribution of extreme events unless there are sufficiently long, historical observation data available. When trained for regression problems, deep learning models are generally trained to predict the conditional mean of a distribution using the mean squared error loss, and thus, fail to capture the tail of the distribution. Extreme events are governed by two parametric distributions [22]: the distribution of block maxima is governed by the generalized extreme value distribution (GEV) and the distribution of excesses over a threshold are governed by the generalized Pareto distribution (GPD).

In this chapter, we propose a novel framework that combines extreme value theory (EVT) with deep learning. Specifically, our framework leverages the strengths of deep learning in modeling complex relationships in geospatio-temporal data as well as the ability of GPD to capture the distribution of excess values with limited observations. However, integrating a deep neural network (DNN) with EVT is a challenge as the loss function minimized by the DNN must be modified to maximize the likelihood function of the GPD. Another computational challenge is that the sufficient statistics of GPD must satisfy certain positivity constraints unlike the output of DNN, which are typically unconstrained. Furthermore, the distribution of extreme values are often temporally correlated. For example, Figure 5.1 shows the relationship between the shape parameter ξ of the GPD distribution for precipitation excesses from one year to the next based on 45-year data from more than 1000 stations considered in our study. This poses a challenge from a modeling perspective as the number of excesses above a threshold tends to vary from one time step to the next. Developing a deep learning approach that can incorporate such variable number of excess

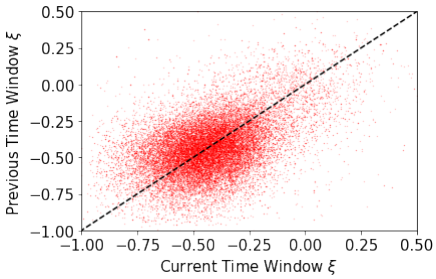


Figure 5.1: Relationship between shape parameter ξ of generalized Pareto distribution for modeling precipitation excesses in two successive time windows.

values as predictors, in addition to other fixed length vectors, is another challenge to be addressed.

The major contributions of this chapter are as follows:

1. We propose a deep learning framework to model the distribution of extreme events. The framework combines CNN with deep sets [111] for modeling geo-spatial relationships among predictors that include fixed-length vectors and variable-sized sets.

2. We propose a re-parameterization method for constraining the outputs of the DNN so that they satisfy the requisite constraints and present an algorithm that learns the GPD parameters in an end-to-end fashion.

3. We evaluate our proposed framework on a real world climate dataset and demonstrate its effectiveness in comparison to conventional EVT and deep learning approaches.

5.2 Related Work

Convolutional neural networks [53] have gained considerable attention over the last several years due to its success in artificial intelligence applications. The strength of CNN lies in its ability to model spatial relationships. For example, CNN has been successfully used to model spatial relationships in geographic applications [46, 57, 68, 82, 94, 113]. A similar intuition motivates the application of CNN to model temporal autocorrelations in time series data. For example, Bai et al. [5] provided empirical evidence that a properly designed convolutional architecture can outperform recurrent neural networks.

More recently, there have been concerted efforts to model both spatial and temporal relationships jointly using CNN. For example, [90], [93], and [43] use 3d convolution to model the spatial and temporal relationships within the layers of their network. Instead of applying convolution to model temporal relationships, some research uses recurrent layers to model temporal relationships [27]. For example, Shi et al. [80] replaces every instance of matrix multiplication in an LSTM with 2d convolution and then feeds the data to the LSTM at each time step. However, none of these approaches are designed for modeling extreme values.

Statistical approaches based on extreme value theory (EVT) [48, 49, 67] are commonly used to infer the distribution of extreme values. Several recent papers have combined deep learning with EVT but often only as a post-processing step. For example,[103] fit a GPD to the residuals of a neural network to help detect cyber risks. Similarly, Yu et al. [109] identify samples with unknown classes at test time using the Weibull distribution. Weng et al. [100] utilize EVT to derive a neural network robustness metric called CLEVER. In none of these cases are deep learning and EVT integrated together within a single end-to-end learning framework. Instead, EVT is used as a post-processing step to identify unusual samples or as a robustness score of the network. In contrast we integrate EVT directly into our deep learning formulation to predict the GPD parameters and training it in an end-to-end fashion. Ding et al. [26] do integrate EVT into the loss function but in an ad-hoc way, where the CDF of the GPD is used to assign weights on extreme samples whose prediction is framed as a binary classification problem. Rather than learning the parameters of GPD, they instead treated them as user-provided hyper-parameters.

5.3 Preliminaries

Let $\mathcal{D} = \left\{ (X_{il}, Y_{il}) \;\middle|\; i \in \{1, \cdots, n\};\ l \in \{1, \cdots, L\} \right\}$ be a geospatio-temporal dataset, where X_{il} denote the predictor attribute values for the time window $(t_{i-1}, t_i]$ in location l and Y_{il} denote the corresponding target (response) values for the time window $(t_i, t_{i+1}]$. Since we are interested in predicting the excesses above a threshold in the next time window, the target variable corresponds to the set of excess values at location l during the period $(t_i, t_{i+1}]$, i.e., $Y_{il} = \{y_{tl} \mid y_{tl} \geq u, t \in (t_i, t_{i+1}]\}$.

In addition, the predictors can be divided into two groups, $X_{il} \equiv (X^v_{il}, X^s_{il})$, where $X^v_{il} \in \mathbb{R}^d$ is a fixed length vector and $X^s_{il} \in \mathbb{R}^{p_i}$ is a variable length vector corresponding to the set of excess values in the previous window, i.e., $X^s_{il} = \{y_{tl} \mid y_{tl} \geq u, t \in (t_{i-1}, t_i]\}$. Note that the number of excess values can vary, e.g., one window may have 10 excess values while the previous window has only 5 excess values. The collections of excess values associated with the current and next time windows form the sets X^s_{il} and Y_{il}, respectively. Our goal is to estimate the conditional distribution $P(Y_{il,j}|X_{il})$ for all the locations l and all windows $i$conditioned on the predictors observed in the current window, where $Y_{il,j}$ is an element of the set Y_{il}.

5.3.1 Extreme Value Theory

This chapter focuses primarily on the use of generalized Pareto distribution (GPD) for modeling the distribution of excesses above a given threshold. For example, in precipitation prediction, one may be interested in modeling the distribution of high precipitation values above a certain threshold.

Let $Y_1, Y_2, \cdots$ be a sequence of independent and identically distributed random variables. Given an excess value $Y = u + y$, where u is some pre-defined threshold, the conditional probability of observing the excess event is:

$$P(Y - u \leq y \mid Y > u) = \begin{cases} 1 - \left[1 + \frac{\xi y}{\sigma}\right]^{-1/\xi}, & \xi \neq 0 \\ 1 - e^{-y}, & \xi = 0 \end{cases}$$

Furthermore, its density function is given by:

$$P(y) = \begin{cases} \frac{1}{\sigma}\left[1 + \frac{\xi y}{\sigma}\right]^{-\frac{1}{\xi}-1}, & \xi \neq 0 \\ \frac{1}{\sigma} e^{-\frac{y}{\sigma}} & \xi = 0 \end{cases} \tag{5.1}$$

subject to the constraint $\forall y : 1 + \frac{\xi y}{\sigma} > 0$. The GPD has two parameters, shape, ξ, and scale, σ. The shape parameter has a significant impact on the overall structure of the probability density. When ξ is negative, the support of the distribution is finite such that $0 < y < -\frac{\sigma}{\xi}$ due to the constraint. When ξ is zero or positive, its support ranges from 0 to positive infinity.

The advantage of using the GPD to model extreme values is its generality as one does not have to know the underlying distribution of the random variable prior to thresholding since the distribution of excesses will be governed by the GPD in relatively general conditions. In many cases, the values of ξ and σ may depend on some contextual features as predictors x. Assuming a linear relationship between ξ and x and between $\log(\sigma)$ and x (the log linear relationship is used to guarantee that the estimate of σ is non-negative):

$$\xi = f_\xi(x) = w_1^T x, \quad \log(\sigma) = f_\sigma(x) = w_2^T x \tag{5.2}$$

where w_1 and w_2 are the model parameters, which can be learned by minimizing the negative log-likelihood of the GPD.

One important consideration when modeling data using a GPD is the choice of threshold u since the threshold must be set high enough for the GPD to be applicable. A common way to evaluate the suitability of a given threshold is by examining the mean residual life plot. If a collection of samples were drawn from a GPD then the empirical distribution of the excesses should have a linear relationship with selected threshold. Specifically, we have:

$$E(Y - u|Y > u) = \frac{\sigma_0 + \xi u}{1 - \xi} \tag{5.3}$$

for threshold u, and $Y \sim GPD(\xi, \sigma_0)$. In the experiment section, we will verify our choice of threshold by examining the mean residual life plot for our precipitation data.

5.3.2 Deep Set

Our framework must be able to handle variable size set of excess values as input predictor, x_{il}^s. To accommodate this, we employ a deep set architecture [111] to transform the variable-length input into fixed size vector. The transformation consists of the following two stages. The first stage is responsible for transforming each element of the set, $x_{il,j}^s$, from its raw representation into a high-level vector representation, $h_{il,j}$ by using a fully connected network, ϕ. These element-wise representations are then aggregated to obtain a fixed-length vector representation for the set. This

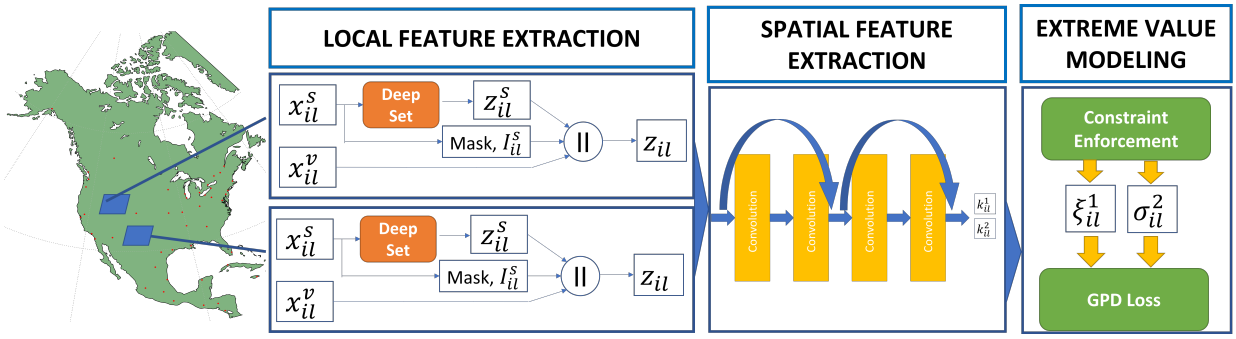


Figure 5.2: Proposed DeepGPD framework.

set-level representation is then used as input to a fully connected network, ρ, to produce the final output representation, $z_{il}^s = \rho\left[\sum_j \phi(x_{il,j}^s)\right]$

5.4 Proposed DeepGPD Framework

Figure 5.2 shows the architecture of our DeepGPD framework, which has the following three major components:

1. **Local Feature Extraction** - This component is responsible for transforming both the (fixed-length) vector-valued, x_{il}^v, and (variable-length) set-valued predictors, x_{il}^s, at each location into a fixed-length feature vector.

2. **Spatial Feature Extraction** - This component models the spatial relationships among the predictors in the data.

3. **Extreme Value Modeling (EVM)** - This component is responsible for ensuring that the constraints on the GPD parameters are satisfied by the induced model.

5.4.1 Local Feature Extraction

This component is responsible for learning a representation of the predictors associated with each location l by utilizing both the set-valued predictors x_{il}^s and the vector-valued predictors x_{il}^v. Learning a representation of the predictors is challenging for two reasons. First, because the set-valued predictors are variable length we must transform them into a fixed length vector so that it

can be used by the later stages of the model. Second, the set-valued predictors may not always be available for some locations.

To address the first challenge, we employ the deep set architecture described in subsection 5.3.2 to transform the set-valued predictors into a fixed-length vector, z^s_{il}.

For the second challenge, there may be some cases where a given grid cell lacks set-valued predictors, x^s_{il}. In these cases we set $z^s_{il} = 0$. However, zeroing the inputs in this way risks the possibility that predictions at locations without set predictors will be distorted. To address this, an indicator variable, I_{il} is introduced to indicate whether set-valued predictors are available at a given location and time. This indicator variable is then concatenated with the vector-valued predictors and the deep set representation of the set-valued predictors to generate the following vector: $z_{il} = z^s_{il} \parallel I_{il} \parallel x^v_{il}$, where $\parallel$ denotes the concatenation operator.

Our deep set implementation assumes there is a maximum set size. Sets that are smaller than this maximum size are padded with dummy values of zero so that each set can be represented by a fixed length vector. Each dummy element is processed in the same way as the real set elements. After each set element is processed by several fully connected layers, only the representations of the actual set elements (i.e. dummy elements excluded) are averaged together. This is implemented through the use of a masking array multiplied by the set member representations element-wise.

5.4.2 Spatial Feature Extraction

After extracting a separate representation for each location, we need to model the spatial relationships between the representations at different locations. `DeepGPD` uses a CNN to capture the geospatial relationships in the data. In our architecture, we arrange the representation extracted from all the gridded locations into a 3-dimensional tensor (excluding the batch dimension) and then provide the tensor as input to a CNN with residual layers [38]. The final linear layer of the CNN produces a response map for each location, $k_{il} \in \mathbb{R}^2$, for the prediction time window $(t_i, t_{i+1}]$.

5.4.3 Extreme Value Modeling (EVM)

The EVM component is designed to predict the conditional distribution of excess values by utilizing the response map generated by the CNN. Specifically, it will convert the CNN output for each location and time window $(t_i, t_{i+1}]$ to the generalized Pareto model parameters, ξ_{il} and σ_{il}. These parameters enable us to infer various statistics about the excess values in the predicted time window, such as the expected values at varying quantiles (including maximum and median value) as well as their return level.

Unlike previous work such as [26], which assumes that ξ and σ are hyperparameters provided by users, `DeepGPD` enables both parameters to be automatically learned from the data. Specifically, the deep architecture is trained to minimize the following negative log-likelihood function of the excess values in the next time step:

$$\mathcal{L}(\{\xi_{il}, \sigma_{il}\}) = \sum_{i,l,j} \left[\log \sigma_{il} + (1 + \frac{1}{\xi_{il}}) \log(1 + \xi_{il} \frac{y_{ilj}}{\sigma_{il}}) \right] \tag{5.4}$$

One major computational challenge in estimating the GPD parameters using a deep learning architecture is the need to enforce positivity constraints on the solution of (5.4) during training. To address this challenge, `DeepGPD` employs a re-parameterization trick to transform (ξ_{il}, σ_{il}) into a pair of unconstrained variables $k_{il} = (k_{il}^{(1)}, k_{il}^{(2)})$ that can be learned by the convolutional neural network.

Theorem 1. *Let* $\{\xi_{il}^*, \sigma_{il}^*\} = \textit{argmin}\ \mathcal{L}(\{\xi_{il}, \sigma_{il}\})$ *subject to the following positivity constraints:*

$$\forall i, j, l: \ \sigma_{il} > 0 \ \textit{and} \ \ 1 + \xi_{il} \frac{y_{ilj}}{\sigma_{il}} > 0$$

By re-parameterizing $(\xi_{il}, \sigma_{il}) \mapsto (k_{il}^{(1)}, k_{il}^{(2)})$ *as follows:*

$$\begin{aligned} \sigma_{il} &= \exp(k_{il}^{(1)}) \\ \xi_{il} &= \exp(k_{il}^{(2)}) - \frac{\exp(k_{il}^{(1)})}{M_{il}} \end{aligned} \tag{5.5}$$

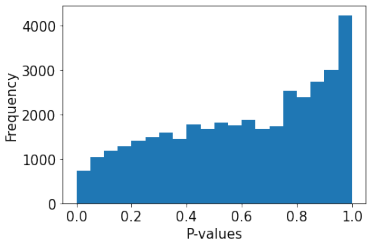


Figure 5.3: Fitted p-Value distribution of K-S test.

and solving for $\{\hat{k}_{il}^{(1)}, \hat{k}_{il}^{(2)}\} = \textit{argmin}\ \hat{\mathcal{L}}(\{u_{il}, v_{il}\})$*, where*

$$\begin{aligned}\hat{\mathcal{L}}(\{u_{il}, v_{il}\}) &= \sum_{ilj}\left[u_{il} + \left(1 + \frac{M_{il}}{M_{il}e^{v_{il}} - e^{u_{il}}}\right)\right. \\ &\left.\times \log\left(1 + e^{v_{il}}\frac{y_{ilj}}{e^{u_{il}}} - \frac{y_{ilj}}{M_{il}}\right)\right] \end{aligned} \tag{5.6}$$

and $M_{il} = \max_j Y_{ilj}$*, then the solution set* $\{\xi_{il}^*, \sigma_{il}^*\}$ *can be derived from the solution for* $\{\hat{k}_{il}^{(1)}, \hat{k}_{il}^{(1)}\}$ *by applying the mapping given in Equation* (5.5)*.*

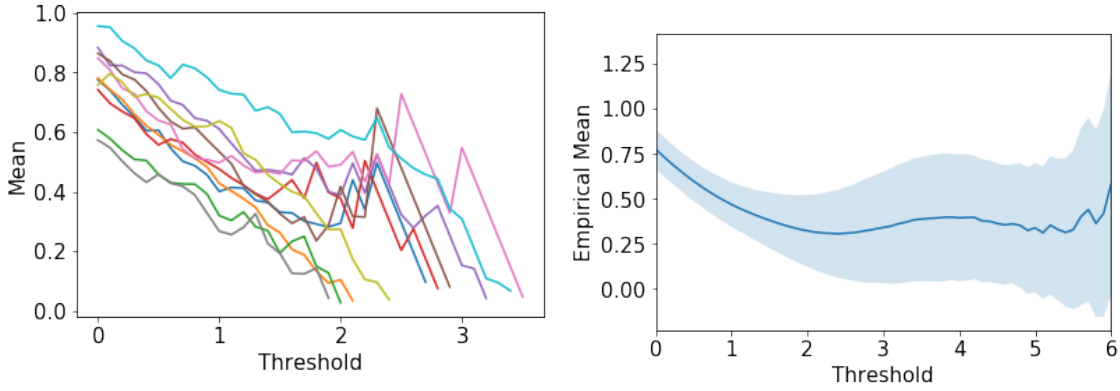
The proof for the preceding theorem can be shown by substituting (5.5) into (5.4), which yields the equivalent objective function for $\hat{\mathcal{L}}(\{k_{il}^{(1)}, k_{il}^{(2)}\})$. Furthermore, since Equation (5.4) can be re-written as follows:

$$\begin{aligned}\sigma_{il} &= e^{k_{il}^{(1)}} \geq 0 \\ 1 + \xi_{il}\frac{y_{ilj}}{\sigma_{il}} &= 1 - \frac{y_{ilj}}{M_{il}} + e^{k_{il}^{(2)}}\frac{y_{ilj}}{e^{k_{il}^{(1)}}} \geq 0\end{aligned}$$

the positivity constraints are automatically satisfied given the fact that $\forall i, l, j : y_{ilj} \leq M_{il}, e^{k_{il}^{(1)}} > 0$ and $e^{k_{il}^{(2)}} > 0$ as long as $k_{il}^{(1)}$ and $k_{il}^{(2)}$ are not equal to $-\infty$.

Corollary 1. *The* `DeepGPD` *framework trained to optimize the loss function in Equation* (5.6) *will generate the maximum likelihood solution for* $\{\xi_{il}^*, \sigma_{il}^*\}$ *in Equation* (5.4) *given the one-to-one mapping with* $\{\hat{k}_{il}^{(1)}, \hat{k}_{il}^{(2)}\}$ *in Equation* (5.5)*.*

The preceding corollary demonstrates the advantages of using our re-parameterization trick to train `DeepGPD` as the values of $\hat{k}_{il}^{(1)}$ and $\hat{k}_{il}^{(2)}$ are less constrained compared to $\{\xi_{il}^*$ and $\sigma_{il}^*\}$.



(a) Excess mean for random samples of locations and time windows. Colors represent different samples.

(b) Excess mean for all locations and time windows.

Figure 5.4: Mean residual life plot for excess precipitation.

This enables the parameters to be more easily learned by `DeepGPD`. All three components of the framework, including deep set and CNN, are trained in an end-to-end fashion using Adam [50]. Once the parameters for $\hat{k}_{il}^{(1)}$ and $\hat{k}_{il}^{(2)}$ are obtained, we can apply Equation (5.5) to derive the corresponding GPD parameters.

5.5 Experimental Results

We evaluate our proposed framework on a 44-year global precipitation data from 1970 to 2013. Specifically, we use daily precipitation values collected from the Global Historical Climatology Network[2] (GHCN) for 1,112 stations located in the Northern Hemisphere (between 22.5°N to 67.5°N) as our target variable. The data is partitioned into 45 non-overlapping one-year time windows. Excess daily precipitation values are considered as any value exceeding one standard deviation above the mean for the station. For predictor variables, we consider the excess values in the previous year as set-valued attributes and the mean and standard deviation of monthly climate values (e.g., convective precipitation rate, solar radiation flux, relative humidity, and sea level pressure) from the NCEP re-analysis project[3] as fixed-length vector-valued attributes.

Our objective is to predict the conditional distribution of the excess precipitation values for

[2]https://www.ncdc.noaa.gov/ghcn-daily-description
[3]https://www.ncep.noaa.gov/

Method	NLL	$\rho(\xi)$	$\rho(\sigma)$
Persistence	0.6271 ± 0.0092	0.40	0.77
Linear Regression	0.6514 ± 0.0114	0.49	0.80
ViT Regression	0.6372 ± 0.0088	0.56	**0.87**
CNN Regression	0.6332± 0.0102	0.41	0.75
Linear GPD	0.6101 ± 0.0035	0.38	0.57
`DeepGPD`	**0.5688** ± 0.0036	**0.57**	0.85

Table 5.1: Comparison between `DeepGPD` against baseline methods in terms of negative log-likelihood (NLL) and correlation (ρ) of predicted ξ and σ to ground truth values.

next year based on the observed excess values and statistics of the NCEP climate variables for the current year. The precipitation data at each location is de-seasonalized separately using its own monthly means and standard deviations. Each 1 year window of predictor and target values are assigned to either training, validation or test sets, with 34 windows in training, 5 in validation and 4 in test. We repeat our experiment 10 times with different train-validation-test splits.

To verify that the excess values follow the GPD, we perform the Kolmogorov-Smirnov goodness of fit test. The KS-test is a non-parametric approach to determine whether a given set of samples is drawn from a given distribution. To do this, we first infer values of the GPD statistics, ξ and σ, from the excess values observed at each location and time window using SciPy genpareto class and then apply the KS-test to assess whether the excess values were indeed drawn from the inferred distribution. We observe that the average p-value over all the locations and time windows is 0.60, which suggests that the inferred distributions accurately fit the data. The distribution of the fitted p-values is shown in Figure 5.3.

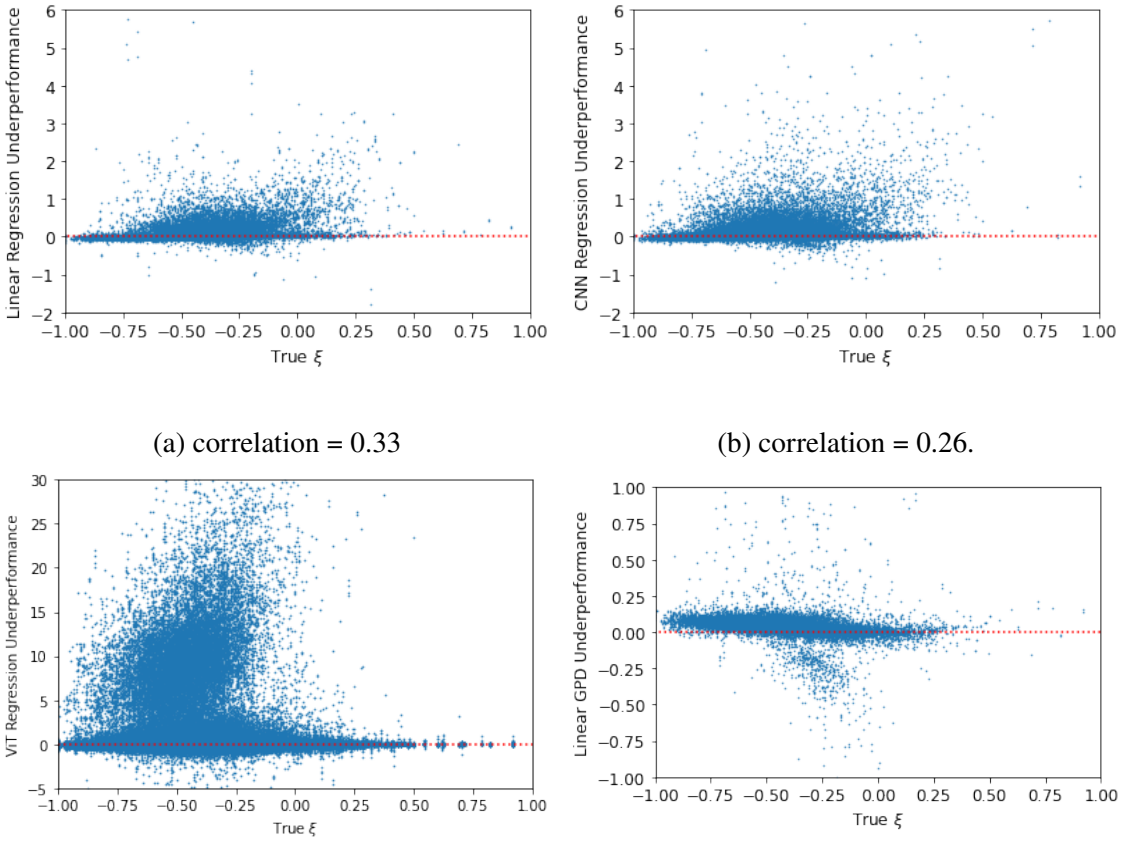
Next we evaluate our choice of excess threshold. Equation 5.3 shows a linear relationship between the choice of threshold and the mean value of the excesses. We empirically verify this linear relationship by plotting the chosen threshold against the mean of excesses in Figure 5.4. The resulting diagram is also known as the mean residual life plot. Ordinarily, the mean residual life plot is used to evaluate the choice of threshold for a single GPD distribution. However, in our case, each window and location has its own GPD distribution. Thus we must evaluate our choice of threshold for this entire family of excess distributions.

Method	Negative log-likelihood
DeepGPD with GHCN only	0.5706 ± 0.0035
DeepGPD with NCEP only	0.5670 ± 0.0040
DeepGPD	0.5688 ± 0.0036

Table 5.2: Results of ablation study.

In Figure 5.4a, we plot a random selection of mean residual life plots at different locations and windows and we find that the relationship between thresholds and the mean residual is approximately linear when the threshold is around 1. In addition, Figure 5.4b shows the average excess mean across all time windows and locations at any given threshold level with shading representing 1 standard deviation in each direction. In the vicinity of 1, the average excess mean across all distributions varies linearly with the choice of threshold and the relatively narrow shade suggests this behavior is shared across most distributions. Notice that the slope of both plots is negative around thresholds of 1 which, based on Equation 5.3, indicates that ξ is negative-valued. Because the relationship between chosen threshold and the empirical mean of the excesses is approximately linear around 1, this justifies our choice of using 1 standard deviation as our excess threshold.

We compare DeepGPD against the following baselines. Similar to DeepGPD, each baseline generates the GPD parameters of a location for the next time window as its output. (1) **Persistence** - A GPD is fitted to the excess values in the current window and used to predict the next window. (2) **Linear Regression** - A linear regression model is trained to predict the GPD parameters (ξ_{il} and σ_{il}) for each location using the NCEP climate variables as well as the fitted GP parameters from previous window as its predictors. (3) **ViT Regression** - This baseline uses the Vision Transformer architecture [28] to predict the GPD parameters and is trained using mean squared error loss. To accommodate our pixel-wise regression problem setting we remove the class token embeddings and set the final MLP to output 2 scalars for each pixel in each patch corresponding to the 2 GPD parameters. (4) **CNN Regression** - This baseline uses the same architecture (including deep set and CNN) and predictors as DeepGPD except it replaces the maximum likelihood loss with a mean square error loss on the GPD parameters, similar to the linear regression baseline. This is similar to the architecture proposed in [38] but consists only of residual layers and a final linear layer



(a) correlation = 0.33

(b) correlation = 0.26.

(c) correlation = 0.01

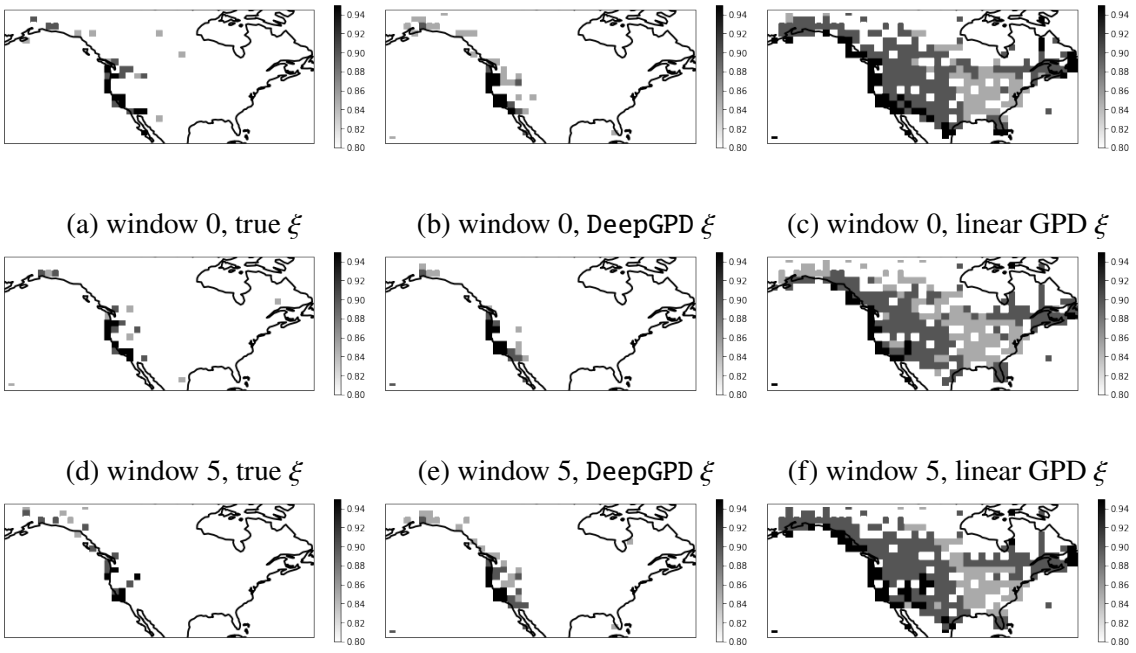
(d) correlation = -0.25

Figure 5.5: Relationship between predictive improvements over baselines and true ξ.

while omitting the pooling and fully connected layers because it performs pixel-wise regression. (5) **Linear GPD** [22] - This is a linear GPD model for predicting the GPD parameters using NCEP and the GPD parameters from the previous window as its predictors (see Equation (5.2)).

5.5.1 Comparison against Baseline Methods

Table 5.1 compares the performance of `DeepGPD` against various baselines using negative log-likelihood and correlation between the predicted and actual ξ values as evaluation metrics. The results in Table 5.1 suggest that, with one exception, `DeepGPD` significantly outperforms all the baseline methods regardless of the metric chosen. The performance of CNN regression and linear regression are poor relative to other baselines. Since both methods employ the mean-square error



(a) window 0, true ξ (b) window 0, DeepGPD ξ (c) window 0, linear GPD ξ

(d) window 5, true ξ (e) window 5, DeepGPD ξ (f) window 5, linear GPD ξ

(g) window 10, true ξ (h) window 10, DeepGPD ξ (i) window 10, linear GPD ξ

Figure 5.6: Comparison between the spatial distribution of the true and predicted ξ values for linear GPD and DeepGPD.

(MSE) of the predicted GPD parameters as its loss function, this shows the importance of explicitly incorporating extreme value theory and its corresponding negative log-likelihood loss to train the model. The only baseline to compare favorably to the proposed method according to any metric is ViT regression which achieves correlations with the ground truth ξ and σ comparable to DeepGPD due to being an extremely expressive model trained explicitly to predict the ground truth ξ and σ values through MSE loss. However, because it doesn't incorporate NLK into its loss in practice it makes poor predictions of the distribution of observed excesses. The relatively strong performance of the persistence method suggests the importance of using information about the excess values in the previous time window to predict their distribution in the next window.

The linear GPD model employs the same loss function (i.e., MLE) as DeepGPD except it uses a linear layer, as opposed to non-linear model, to learn the mapping from the predictors to the GPD parameters. This has two additional implications. First, linear GPD is unable to directly incorporate the set-valued predictors of the GHCN; therefore they can only use the inferred GPD

statistics from previous window as one of its predictors. Furthermore, existing linear GPD approach also does not incorporate spatial information since it does not include a spatial component such as CNN. As a result, the proposed DeepGPD method outperforms linear GPD by a significantly large margin, demonstrating the importance of non-linearity and incorporating spatial relationship into the modeling task. Nevertheless, the linear GPD still outperforms other baselines, suggesting the value of incorporating GPD into the learning formulation.

Table 5.2 compares our full model against variations that utilize only the vector-valued (NCEP only) or the set-valued predictors (GHCN only). The results show that all three methods achieve comparable performance with significant overlap in their confidence intervals. This suggests that there exists large amount of redundant information between both types of predictors. Although our model can effectively utilize the set-valued or vector-valued predictors, leveraging them together does not appear to improve the model.

5.5.2 Distribution of Estimated GPD Parameters

The previous subsection compares the overall performance of DeepGPD against the various baseline methods in terms of their negative log-likelihood and correlation with the ground truth GPD shape parameter values. Table 5.1 indeed shows that the proposed method significantly outperforms the baseline methods but this raises the question of the reason for its performance improvement. Since the results in Table 5.1 are based on an aggregation over multiple time windows and locations, we need to analyze the relative performance of the various baseline methods compared to DeepGPD at a finer level. To do this, we first compute the difference between the negative log-likelihood (NLL) of each baseline against the proposed method for each location and time window. If the difference is positive, this suggests that the NLL value of the baseline is worse (i.e., higher) than the NLL of DeepGPD. Figure 5.5 displays the relationship between the NLL difference of each baseline relative to the proposed method and the true GPD shape parameter ξ. First, note that there is a positive bias along the y-axis in the plots, which suggests that the performance improvement in DeepGPD is observed for the majority of the locations and time windows of our data. In fact, DeepGPD

outperforms the baselines in 58% to 89% of locations and windows. Second, in the case of linear and CNN regression-based methods, observe that there is a positive correlation between the value of ξ and relatively stronger performance by the proposed method. This suggests that `DeepGPD` performs best in situations where the tails of the distribution are heaviest. Since extreme events are the ones most important to predict, the strong performance of `DeepGPD` in these scenarios is a major point in its favor. This plot also illustrates that there are a considerable number of samples for which the ViT Regression model makes much worse predictions than the proposed model. The plot also suggests that linear GPD outperforms `DeepGPD` when the ξ parameter is in the range between -0.5 and 0. Nevertheless, there are still more data points with positive NLL difference for the same range of ξ values.

Next, we examine the spatial distribution of the predicted ξ values and compare them to their ground truth distribution. Since large values of ξ are especially important we focus on them. In Figure 5.6 we plot the predicted values of ξ for three time windows with grid cells colored based on their relationship to certain high thresholds. These thresholds range from 85th to 95th quantiles calculated based on the ground truth data with ξ values exceeding the 95th quantile of the ground truth ξ values colored black and everything below the 80th percentile colored white and any ξ in between colored gray. We produce separate plots for the ground truth, proposed method, and the linear GPD baseline. Due to space limitations we only show the results for linear GPD because it is best NLL performance among the baseline methods. These plots show that the proposed method does well in predicting the general spatial distribution of the highest ξ values by identifying which locations have relatively high or low ξ values. The worst predictions come from the linear GPD baseline which struggles to capture the variability of the data with almost all locations visualized as black or gray due to the large positive bias of the model as well as the low standard deviation of its predictions.

5.6 Conclusions

In this chapter we identified the limitations of existing deep learning methods in predicting the distributions of extreme values. To address this limitation we proposed a novel deep learning architecture capable of learning the parameters of the generalized Pareto distribution while satisfying the conditions placed on those parameters. We evaluated our results on a real world climate data set and showed that our proposed framework outperformed various baseline methods.

CHAPTER 6

DEEP EXTREME MIXTURE MODEL FOR GEO-SPATIO-TEMPORAL DATA

6.1 Introduction [1]

The prediction of extreme events encompasses many important applications including the prediction of heat waves [79], equity risk [30], corrosion phenomena [78], and flooding [13]. There are many ways to formalize the notion of an extreme event [21]. In this work, we will focus on extreme events that correspond to values that exceed some fixed threshold. The previous chapter presents a deep learning framework called *DeepGPD* for inferring the future distribution of extreme events based on the historical incidence and other predictors. Such a prediction task is particularly useful for long-term forecasting applications such as projecting the future climate change scenarios at a location. Nevertheless, there are other applications that require accurate point estimate of the forecasts at each time step to enable determination of the magnitude, timing, and frequency of the extreme events. As shown in previous works [1, 65], regression methods designed to learn the future distribution of data may not necessarily have low mean square prediction error in terms of their point-wise estimation. A new deep learning formulation for geo-spatio-temporal applications is therefore needed to enable accurate point-wise forecasts while preserving the properties of the inferred distribution.

Attempting to simultaneously predict the distribution, timing, and frequency of extreme events in addition to making accurate point predictions imposes three challenges. The first challenge is that the statistical distributions used to model extreme events have natural constraints on their model parameters. These constraints must be preserved to ensure its fidelity in terms of characterizing the tail distribution of a random variable. Second, when modeling the predictive distribution of a target variable, it is common to use the expected value (mean) of the distribution as its point

[1] **This chapter is adapted from an unpublished paper: Tyler Wilson, Pang-Ning Tan, Lifeng Luo, Andrew McDonald, Asadullah Galib. "DEMM: Deep Extreme Mixture Model".**

estimate. However, this will require modeling the full distribution of the target variable, not just its excesses over a threshold, which is the approach used in *DeepGPD*. Additionally, the mean of the distribution that governs the extreme values is only well-defined if the model parameters satisfy certain feasibility conditions. This imposes additional constraints on the optimizing algorithm for learning the parameters. Finally, the prediction results may vary depending on the threshold used to define the extreme events. In general, a prediction model trained on a specific threshold of excess values may not necessarily perform well when applied to a different threshold. Since the true excess threshold is often unknown, it is possible that users may want to vary the threshold when applying the model. Thus, developing a robust model with the flexibility to alter the threshold that defines extreme values at test time without requiring the model to be retrained will be valuable for users.

When predicting extreme values it is important to consider other features of the data as well. For example, the geo-spatio-temporal data can be zero inflated as in precipitation prediction. A typical way to model these zero inflated distributions is a hurdle model[24], which uses separate random variables to model the distribution of zero and non-zero values. However, this approach fails to account for extreme values. By incorporating extreme value theory into the model we hope to develop a better specified model especially in applications where extreme values are particularly important.

Developing a deep learning framework for predicting geo-spatio-temporal extreme events will require addressing all of these challenges. In this chapter, we propose a framework called Deep Extreme Mixture Model (*DEMM*) that fuses a deep learning based hurdle model with extreme value theory (EVT) to predict the future values of a geo-spatio-temporal variable as well as the conditional distribution of its extreme values. The core of the framework is a novel, deep-learning based hurdle model. A typical hurdle model is a mixture model with two underlying components—one component governing the strictly zero values while the other component governing the distribution of its non-zero values. Instead, *DEMM* incorporates a third component that uses the generalized Pareto distribution (GPD) to model the distribution of extreme values above a specified threshold. The parameters of the hurdle model in *DEMM* are inferred using a 3-d convolutional neural network

(CNN), which is capable of capturing both the spatial and temporal relationships within the data.

In summary, the contributions of this chapter are:

1. We propose a novel architecture capable of predicting the timing, frequency, and distribution of geo-spatio-temporal extreme events while simultaneously making point predictions.

2. We propose a novel re-parameterization which ensures that the standard set of constraints on the GPD are satisfied and that it has a well defined mean.

3. We propose a technique for allowing the user to dynamically alter their chosen extreme value threshold at test time without retraining the model.

4. We demonstrate the effectiveness of the *DEMM* in predicting the timing and frequency of extreme events on a real world precipitation data set.

The remainder of this chapter is organized as follows: First, in Section 6.2, we will describe recent literature related to the use of neural networks to estimate the parameters of mixture models. Section 6.3 introduces a formal statement of the prediction problem and provides a brief introduction to extreme value theory and hurdle model. Section 6.4 describes the underlying components of the proposed *DEMM* architecture. Section 6.5 presents the experimental results, followed by concluding remarks in Section 6.6.

6.2 Related Work

This section reviews recent work using deep learning to infer the parameters of mixture models including hurdle models and the distributions studied in extreme value theory. An early use of neural networks to predict the parameters of a mixture distribution comes from Bishop [9] which proposes "Mixture Density Networks" which combines neural networks with a gaussian mixture model to represent what are in principal arbitrary conditional distributions. Though their model is not trained to make point predictions they show how point estimates can be derived from their conditional distributions. More recently, deep neural networks have been integrated with mixture models for a wider range of purposes. Zong et al. [117] propose a deep gaussian mixture model

for unsupervised anomaly detection. And Viroli and McLachlan [98] propose a deep gaussian mixture model for clustering. Hurdle models were originally proposed by Cragg [24] and some recent work has used deep neural networks to estimate the parameters of hurdle models. Kong et al. [52] propose a deep hurdle network for multi-target regression. Vandal et al. [95] use a deep neural network to estimate the parameters of a hurdle model using a log-normal distribution as its second component. Additionally, they use monte-carlo dropout to account for uncertainty in model parameters. In Bacry et al. [4] a multinomial based zero inflated model is proposed and applied to medical data.

Previously, traditional statistical approaches have been used to infer the distribution of extreme values [48, 49, 67]. However, these statistical approaches for incorporating extreme value theory generally assume there is a relatively simple relationship between predictors and the generalized Pareto statistical parameters. In addition, this previous work generally does not model the full distribution of the data but instead only focuses on the distribution of extremes and generally does not incorporate point prediction. The previous work combining deep learning with extreme value theory does not generally tightly couple the two techniques. Instead, extreme value theory is used as a post-processing step [100, 109] or utilized in a limited or ad hoc way [26]. To the best of our knowledge, the previous *DeepGPD* framework described in Chapter 5 is the only work exploring the use of deep learning to infer the parameters of distributions studied in extreme value theory.

6.3 Preliminaries

This section formalizes the problem statement and provides a brief introduction to extreme value theory and the hurdle model, both of which are both integral to the proposed *DEMM* framework.

6.3.1 Problem Statement

Let $\mathcal{D} = \left\{ (X_{lw}, Y_{lw}) \;\middle|\; w \in \{1, \cdots, n\};\ l \in \{1, \cdots, L\} \right\}$ be a geo-spatio-temporal dataset, where $X_{lw} \in \mathbb{R}^{d \times \tau}$ denotes the sequence of d predictors for a prediction window w of width τ at location l while $Y_{lw} \in \mathbb{R}$ denotes the scalar value of the target variable associated with the prediction window.

The locations are assumed to be organized onto a spatial grid $\mathcal{S}$ while each prediction window w is associated with a series of discrete time steps, $[t_1^w, t_2^w, \cdots, t_\tau^w]$. Thus, each $X_{lwi} \in \mathbb{R}^d$ represents a vector of predictors at a particular location l at time step $t_i^{(w)}$ while Y_{lw} represents the observed value at time t_τ^w. For brevity, let $X_{:wi} \in \mathbb{R}^{L \times d}$ denote a gridded snapshot image of the predictors across all spatial locations at time step $t_i^{(w)} \in w$ while $X_{:w} = \langle X_{:wi} \;\middle|\; i \in \{1, \cdots, \tau\} \rangle$ denotes a sequence of such snapshots. Similarly, the gridded snapshot of the target variable at time step $t_\tau^{(w)}$ will be denoted as $Y_{:w}$. To determine whether the value of the target variable is an extreme value, let U_{lw} be the excess value threshold at a given location l and prediction window w. Specifically, any samples above this threshold will be considered extreme values. The set of threshold values for all locations in a given prediction window w is denoted as $U_{:w}$. Note that in practice users will likely assign U_{lw} a constant value at all prediction windows and possibly all locations but for full generality we allow it to be defined separately at each prediction window and location.

6.3.2 Extreme Value Theory

There are two widely-used statistical distributions for modeling extremes—(1) the generalized extreme value (GEV) distribution, which is used to model the distribution of block maxima, and (2) the generalized Pareto distribution (GPD), which is used to model the distribution of excesses above a given threshold. Since we are primarily interested in the distributions of excesses over thresholds, this chapter will focus only on the generalized Pareto distribution.

A brief introduction to the GPD is given in Section 5.3.1. Recall that the density function of GPD is given by

$$P(y) = \begin{cases} \frac{1}{\sigma}\left[1 + \frac{\xi y}{\sigma}\right]^{-\frac{1}{\xi}-1}, & \xi \neq 0 \\ \frac{1}{\sigma} e^{-\frac{y}{\sigma}} & \xi = 0 \end{cases} \tag{6.1}$$

Observe that the distribution is characterized by two parameters: shape, ξ, and scale, σ. Furthermore, there are two key constraints that must be satisfied by the GPD parameters, namely, a positivity constraint on its scale parameter σ and a more complex constraint involving the shape,

scale, and samples from the distribution, i.e.:

$$\textbf{GPD constraints:} \qquad \sigma > 0 \quad \text{and} \quad \forall y : \; 1 + \frac{\xi y}{\sigma} > 0 \tag{6.2}$$

Note that this second constraint is always satisfied when $\xi \geq 0$ since $y > 0$. Another important fact about the GPD distribution is that its expected value is given by $E[Y] = \frac{\sigma}{1-\xi}$ when $\xi < 1$ but is undefined otherwise.

6.3.3 Hurdle Model

In some cases a random variable may have a large number of zeros. When modeling daily precipitation, for instance, most days of zero rainfall. In these cases it may be helpful to use a hurdle model separately model the probability that the variable is 0 and its probability distribution when non-zero. The hurdle model is essentially a mixture model consisting of two components. The first component has a constant value of 0 while the second component governs the distribution of non-zero values. This can be represented formally as follows:

$$P(Y = y) = \begin{cases} p & y = 0 \\ (1 - p) \times f_Y(y) & y > 0 \end{cases}$$

where Y is the random variable modeled by the hurdle model, p is its probability of being 0 and f_Y is the probability density function of Y when its value is non-zero. Any valid probability density function can be used as the second component of the hurdle model (i.e. f) as long as its integral over y from 0 to infinity is 1 since:

$$\begin{aligned} \int_0^\infty P(y)dy &= \int_0^\infty \Big(pI[y = 0] + (1 - p) f_Y(y) I[y \neq 0] \Big) dy \\ &= p + (1 - p) \int_0^\infty f_Y(y) I[y \neq 0] dy \\ &= p + (1 - p) \int_0^\infty f_Y(y) dy \end{aligned}$$

where $I[\cdot]$ denotes the indicator function. Furthermore, since Y is assumed to be a continuous random variable, the density function f_Y is zero at any given value of y, including $y = 0$. Hence $\int_0^\infty P(y)dy = 1$ as long as $\int_0^\infty f_Y(y)dy = 1$.

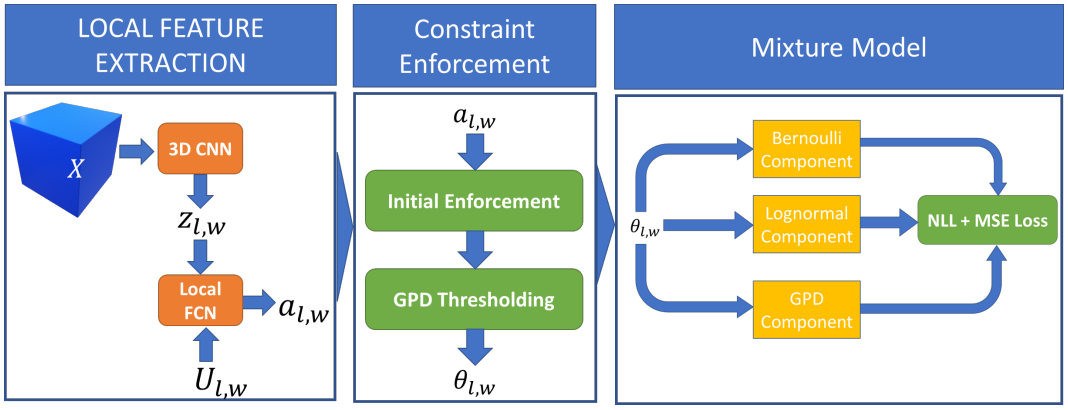


Figure 6.1: An overview of the proposed *DEMM* architecture

6.4 Deep Extreme Mixture Model

The core of the Deep Extreme Mixture Model (*DEMM*) is a mixture model which governs the conditional distribution of the target variable, Y_{lw}. Figure 6.1 presents a schematic illustration of the *DEMM* architecture, which can be divided into three major components. The first component is a 3-d convolutional neural network, which is responsible for modeling the spatio-temporal relationships within the predictors in addition to inferring the impact of the choice of threshold on the overall distribution. The second component is a constraint enforcement module, which is responsible for transforming the output of the neural network, a_{lw}, into a feasible set of mixture model parameters, θ_{lw}. The third component corresponds to the mixture model itself. We will introduce the mixture model at the heart of the *DEMM* first before describing the rest of the components in detail.

6.4.1 Mixture Model

The *DEMM* is centered around estimating the parameters of a mixture model. This mixture model is a combination of 3 probability functions, each of which is responsible for a different range of values for the target variable. The three components of the mixture model combined have a total of 6 parameters, which is unique for each window w and location l.

Because the model is intended for use with zero inflated data, such as precipitation, it is based

on a hurdle model, extended to account for the modeling of extreme values. The first component of the mixture model is a probability function that the target variable has the value of zero. Since the variable of interest is assumed to be non-negative, this component corresponds to the lower boundary of the distribution.

The second component governs the distribution of non-zero values below a certain threshold, U_{lw}. For precipitation prediction, a truncated log-normal distribution with parameters μ_{lw} and s_{lw} can be used, though the *DEMM* framework can accommodate other types of density functions. The density function of a non-truncated log-normal distribution with parameters μ_{lw} and s_{lw} is given by:

$$\hat{f}_1(Y_{lw}; \mu_{lw}, s_{lw}) = \frac{1}{Y_{lw}\sigma_{lw}\sqrt{2\pi}} \exp - \frac{(\log Y_{lw} - \mu_{lw})^2}{2\sigma_{lw}^2} \tag{6.3}$$

Let $\hat{F}_1$ be the cumulative distribution function (cdf) of $\hat{f}_1$. The truncated log-normal distribution function can thus be expressed as follows:

$$f_1(Y_{lw}) = \frac{\hat{f}_1(Y_{lw})}{\hat{F}_1(U_{lw}; \mu_{lw}, s_{lw})} \tag{6.4}$$

with the domain $0 < Y_{lw} < U_{lw}$. Note that the subscript 1 here is used to denote the second component of the mixture model.

Together, the first two components of the mixture model are quite similar to a hurdle model. However, the choice of the second component of the hurdle model will have a significant impact on how well the mixture model fits the empirical distribution. However, there is strong theoretical motivation for choosing the generalized Pareto distribution for sufficiently large U_{lw} [22]. Thus, the generalized Pareto distribution can be used as the third component of the mixture model and it will govern the distribution of excesses over a threshold. This ensures that the model is well specified for large values of Y_{lw} which exceed U_{lw}. With parameters ξ_{lw} and σ_{lw} its density function is given in Equation 6.1 and is denoted f_2.

To ensure that its integral over the domain of Y_{lw} is equal to 1, the last two components underlying the mixture model must be rescaled. The lognormal component is rescaled by a factor of $(1 - p_{lw}^{(0)}) \times p_{lw}^{(1)}$, where $p_{lw}^{(0)}$ represents the probability that $Y_{lw} = 0$ and $p_{lw}^{(1)}$ represents the

probability it is non-zero and will not exceed the threshold. The GPD component must be rescaled by a factor of $(1 - p_{lw}^{(0)}) \times (1 - p_{lw}^{(1)})$. Thus, the full distribution of the mixture model used in *DEMM* is:

$$P(Y_{lw} \big| X_{:w}; U_{:w}; \theta_{lw}) = \begin{cases} p_{lw}^{(0)} & Y_{lw} = 0 \\ (1 - p_{lw}^{(0)}) \times p_{lw}^{(1)} \times f_1(Y_{lw}; \mu_{lw}, s_{lw}) & 0 < Y_{lw} < U_{lw} \\ (1 - p_{lw}^{(0)}) \times (1 - p_{lw}^{(1)}) \times f_2(Y_{lw}; \xi_{lw}, \sigma_{lw}, U_{lw}) & U_{lw} < Y_{lw} \end{cases}$$

Collectively, the parameters of the mixture model are denoted as the following 6-dimensional vector:

$$\theta_{lw} = (p_{lw}^{(0)}, p_{lw}^{(1)}, \mu_{lw}, s_{lw}, \xi_{lw}, \sigma_{lw}) \tag{6.5}$$

The target variable is a sample from the conditional distribution defined by this mixture model. Given the mixture model parameters, it is easy to compute the negative log likelihood loss as:

$$\begin{aligned} \mathbb{L}_{\text{NLL}} = -\sum_{lw} \Big\{ & I[Y_{lw} = 0] \times \log(p_{lw}^{(0)}) \\ + & I[0 < Y_{lw} < U_{lw}] \times [\log(1 - p_{lw}^{(0)}) + \log(p_{lw}^{(1)}) + \log(f_1(Y_{lw}; \mu_{lw}, s_{lw}))] \\ + & I[U_{lw} < Y_{lw}] \times [\log(1 - p_{lw}^{(0)}) + \log(1 - p_{lw}^{(1)}) + \log(f_2(Y_{lw}; \xi_{lw}, \sigma_{lw}, U_{lw}))] \Big\} \end{aligned} \tag{6.6}$$

In addition the expected value of the mixture model can be easily computed as a weighted sum of the component means:

$$\begin{aligned} \hat{Y}_{lw} = & \; p_{lw}^{(0)} \times 0 \\ + & \; (1 - p_{lw}^{(0)}) \times (p_{lw}^{(1)}) \times \exp(\mu_{lw} + s_{lw}^2/2) \times \frac{\Phi(\frac{\ln(U_{lw}) - \mu_{lw} - s_{lw}^2}{\sigma_{lw}})}{\Phi(\frac{\ln(U_{lw}) - \mu_{lw}}{\sigma_{lw}})} \\ + & \; (1 - p_{lw}^{(0)}) \times (1 - p_{lw}^{(1)}) \times (\frac{\sigma_{lw}}{1 - \xi_{lw}}) \end{aligned}$$

where Φ is the cumulative distribution function of a standard normal distribution. The value $\hat{Y}_{lw}$ can be used as a point prediction Y_{lw}.

6.4.2 Deep Neural Network

This section introduces the deep neural network architecture that forms the first major component of the *DEMM* shown in Figure 6.1. The neural network is responsible for modeling spatio-temporal relationships among the predictors as well modeling the influence of threshold choice on the mixture model parameters. Thus, it takes as input a batch of $X_{:w}$ and $U_{:w}$. The activations from its final layer are then passed on to the constraint enforcement module. The mapping from the predictors and thresholds for a particular window to final activations can be represented formally as:

$$a_{:w} = \text{CNN}(X_{:w}, U_{:w}) \tag{6.7}$$

where $a_{:w} \in \mathbb{R}^{L\times 6}$ is the activations of the final layer of the neural network associated with a given window w. Since the final activations for each location and window will ultimately be mapped to the mixture model parameters, therefore, $a_{lw} \in \mathbb{R}^6$ to match the dimensionality of θ_{lw}.

The neural network can be further divided into a spatio-temporal feature extraction module and a local feature extraction module. The 3-d CNN is the basis for the spatio-temporal feature extractor. 3-d CNNs [33] are generalizations of the 2-d convolutions conventionally used in image datasets to a 3rd dimension. In the context of this chapter, the 3 dimensions correspond to latitude, longitude, and time. The 3-d CNN works by computing the inner product between a filter of parameters with small localized regions within the 3-d spatio-temporal volume. These convolutional layers alternate with non-linear activation functions such as the ReLU or tanh functions. Similar 3-d CNNs have had success in other spatio-temporal applications [15, 44, 74, 91, 92]. The application of the 3d CNN to the predictors can be written formally as $z_{:w} = g(X_{:w})$ where $z_{:w} \in R^{L\times d}$ and g is the non-linear function associated with the trained 3-d CNN.

Once the spatio-temporal features have been extracted, they need to be combined with the chosen threshold at each location so that the parameters of the mixture model can be estimated. This is accomplished by concatenating the threshold at each location, U_{lw}, to the location's spatio-temporal feature representation, z_{lw}, and feeding the results to a fully connected neural network (FCN). The output activation of the network is $a_{lw} = h(z_{lw}, U_{lw})$, where h represents the non-linear function

associated with the fully connected neural network. Observe that each location and window are processed separately by the fully connected neural network.

6.4.3 Constraint Enforcement

Since the final output of the neural network is completely unconstrained, they may not be suitable for use as the parameters of the mixture model, which must satisfy certain feasibility conditions including the GPD inequality constraints stated in Equation (6.2). Specifically, $p_{lw}^{(0)}$ and $p_{lw}^{(1)}$ are constrained to be between 0 and 1, s_{lw} and σ_{lw} are constrained to be non-negative, while ξ_{lw} and σ_{lw} must jointly satisfy:

$$\forall Y_{lw} : 1 + \frac{\xi_{lw} Y_{lw}}{\sigma_{lw}} > 0 \tag{6.8}$$

Given that the mean of the mixture model, $\hat{Y}_{lw}$, will be used as a point estimate of Y_{lw}, this requires computing the mean of the 3 components of the mixture model. However, the mean of the GPD distribution is only well-defined when $\xi_{lw} < 1$. This imposes another constraint that needs to be satisfied.

The constraint enforcement module will transform the output activation of the neural network, a_{lw}, into parameters of the mixture model, θ_{lw}, such that all the constraints are satisfied. Formally, the transformation can be stated as applying the following mapping function, $c : \mathbb{R}^6 \rightarrow \mathbb{R}^6$:

$$\theta_{lw} = c(a_{lw}) \tag{6.9}$$

The constraints on $p_{lw}^{(0)}$ and $p_{lw}^{(1)}$ are easy to achieve by passing the corresponding activations through a sigmoid function, $\sigma[\cdot]$, i.e.:

$$p_{lw}^{(0)} = \sigma[a_{lw}^{(1)}] = \frac{1}{1 + e^{-a_{lw}^{(0)}}}, \qquad p_{lw}^{(1)} = \sigma[a_{lw}^{(1)}] = \frac{1}{1 + e^{-a_{lw}^{(2)}}}$$

The non-negativity constraints on s_{lw} and σ_{lw} are similarly easy to achieve by passing the activations through the exponential function, as shown below:

$$s_{lw} = \exp[a_{lw}^{(2)}], \qquad \sigma_{lw} = \exp[a_{lw}^{(3)}]$$

More challenging however is enforcing the constraints involving the shape parameter of the GPD distribution, ξ_{lw}.

6.4.3.1 GPD Constraints

Recall that there are two constraints on the GPD shape parameter:

$$\begin{aligned} \forall Y_{lw} : 1 + \frac{\xi_{lw} Y_{lw}}{\sigma_{lw}} > 0 \\ \xi_{lw} < 1 \end{aligned} \tag{6.10}$$

where the second constraint is to ensure that the mean of the GPD, and thus the mixture model, is well-defined. Our approach for ensuring both constraints are satisfied will proceed in three steps. First a base GPD constrainer function is applied to ensure that ξ satisfies the first constraint. Next, a shifted softplus function is used to ensure that $\xi_{lw} < 1$. Finally, a gated thresholding function will be applied to ensure that the base GPD constrainer and shifted softplus function work appropriately together so that both constraints involving the shape parameter are simultaneously satisfied. We will discuss each of these steps to enforce the shape parameter constraint in order.

As mentioned above, the base GPD constrainer will ensure that the first GPD constraint is satisfied. Let $a_{lw}^{(4)}$ and $a_{lw}^{(5)}$ be the neural network activations corresponding to the GPD parameters and m is the supremum, i.e., least upper bound, of Y_{lw}. Define $\sigma_{lw} = \exp(a_{lw}^{(5)})$, and define:

$$\begin{aligned} \hat{\xi}_{lw} &= c_\xi[a_{lw}^{(4)}, a_{lw}^{(5)}] \\ &= [\exp(a_{lw}^{(4)}) - 1] \times \exp(a_{lw}^{(5)})/(m + \epsilon) \end{aligned} \tag{6.11}$$

where c_ξ is the base GPD constrainer function. The initial output of the base GPD constrainer is denoted as $\hat{\xi}_{lw}$ rather than ξ_{lw} to indicate that its output must be further constrained to ensure that the second constraint (i.e. $\xi_{lw} < 1$) is satisfied.

The second constraint $\xi_{lw} < 1$ will be enforced using the shifted softplus function and the gated thresholding function. The shifted softplus function is defined as:

$$S(\xi_{lw}) = (1 - \epsilon) - \frac{1}{\beta} \log \left[1 + \exp[(1 - \epsilon - \xi_{lw}) \times \beta] \right] \tag{6.12}$$

where β is a hyper-parameter (set to 10 for this work), and ϵ is a small positive value (set to 0.05 for this work). The shifted softplus function is a shifted and rotated version of the softplus function. It is easy to verify that $\lim_{\hat{\xi}_{lw} \to \inf} S(\hat{\xi}_{lw}) = (1 - \epsilon)$ and $\lim_{\hat{\xi}_{lw} \to -\inf} S(\hat{\xi}_{lw}) = \hat{\xi}_{lw}$.

Note that the general outcome of applying the shifted softplus function is to reduce the value of its input so that $S(\hat{\xi}_{lw}) < \hat{\xi}_{lw}$. When $\hat{\xi}_{lw} > 0$ this is no problem since the only constraint ξ_{lw} needs to satisfy is $\xi_{lw} < 1$, but when $\hat{\xi}_{lw} < 0$, this may result in a situation where the first shape constraint now becomes violated so when $\hat{\xi}_{lw} < 0$, S risks violating the other constraint. This is avoided using the gated thresholding function, T defined as:

$$T(\hat{\xi}_{lw}) = v(\hat{\xi}_{lw}) \times S(\hat{\xi}_{lw}) + (1 - v(\hat{\xi}_{lw})) \times \hat{\xi}_{lw} \tag{6.13}$$

where,

$$v(\hat{xi}_{lw}) = \begin{cases} 0 & \hat{\xi}_{lw} < 0 \\ \hat{\xi}_{lw}/(1-\epsilon) & 0 < \hat{\xi}_{lw} < 1 - \epsilon \\ 1 & 1 - \epsilon < \hat{\xi}_{lw} \end{cases}$$

The basic idea of the gated thresholding function is that when its input, $\hat{\xi}$, is less than 0, then its input will be returned unchanged. However, when the input $\hat{\xi}$ is greater than $1 - \epsilon$, the shifted softplus function is used to reduce its value to be less than 1. When $\hat{\xi}$ is between 0 and $1 - \epsilon$, it will smoothly interpolate between the identity function and shifted softplus function to ensure continuity. This results in a function that will constrain $\xi_{lw} < 1$ while also ensuring that its output satisfies the GPD constraints as long as the input does.

We thus have:

$$\begin{aligned} \theta_{lw} &= c(a_{lw}) \\ &= c\left((a_{lw}^{(0)}, a_{lw}^{(1)}, a_{lw}^{(2)} a_{lw}^{(3)}, a_{lw}^{(0)}, a_{lw}^{(4)}, a_{lw}^{(5)})\right) \\ &= \left(\sigma(a_{lw}^{(0)}), \sigma(a_{lw}^{(1)}), a_{lw}^{(2)}, \exp(a_{lw}^{(3)}), T\left(c_{\xi}(a_{lw}^{(4)}, a_{lw}^{(5)})\right), \exp(a_{lw}^{(5)})\right) \end{aligned} \tag{6.14}$$

6.4.4 Training

The model is trained end to end using a combination of negative log likelihood and mean squared error. Specifically *DEMM* is trained to minimize the following objective function:

$$\mathbb{L}_{\text{NLL}} + \lambda \sum_{l,w} (Y_{lw} - \hat{Y}_{lw})^2 \tag{6.15}$$

where $\mathbb{L}_{\text{NLL}}$ denote the negative log-likelihood function given in (6.6) and λ is a hyper-parameter representing the tradeoff between minimizing the negative log-likelihood and mean squared error loss. One challenge when training the model is choosing the appropriate value for the threshold U_{lw} at each location and time window. To provide more flexibility and allow users to chose any reasonable threshold at test time, during training U_{lw} is sampled uniformly at random from the interval (0.5, 0.95). In principle the range from which the threshold is randomly selected could be extended. This ensures that at test time any threshold from this interval should be usable without retraining the model. Finally, the *DEMM* framework is trained end-to-end using Adam [50].

6.5 Experimental Evaluation

6.5.1 Data

We evaluate our model on a real world precipitation data set drawn from two sources. Predictors are precipitation forecasts from the SubX project[2]. Specifically, an 11-member ensemble of daily precipitation forecasts are generated every week by a numerical model for each location for the next 35 days (i.e $X_{lwi} \in \mathbb{R}^{11}, i \in \{1, 2, \cdots, 35\}$). We compute the rolling 3-day average of each ensemble member. For our target we use precipitation observations from NLDAS-2, specifically the average observed precipitation at each location 10-12 days in advance. We limit our experiments to the continental United States, at a 1 degree resolution, and the years from 1999-2020. The predictors are log transformed and standardized.

6.5.2 Baseline Methods

We compare the proposed *DEMM* framework against the following baseline methods:

1. Ensemble Mean - This approach uses the average value of the ensemble members for days 10-12 as its point prediction.

[2]http://cola.gmu.edu/subx/

2. Kernel Density Estimation - Since the ensemble mean only provides a point estimate, to obtain the density function of the ensemble member forecasts, we apply kernel density estimation to predict the distribution of extreme events from the ensemble members. Specifically, we use a Gaussian kernel centered at each ensemble member forecast, with the kernel bandwidth treated as a hyper-parameter.

3. Lognormal Hurdle Model - This method is similar to the proposed framework except it omits the generalized Pareto distribution from the mixture model.

6.5.3 Experimental Setup

Hyper-parameters were selected using grid search. The learning rates varied in the range from 1×10^{-4} to 1×10^{-2}. The hidden dimension varied in the range from 10 to 40. The parameter controlling the tradeoff between NLL and MSE (i.e. λ) varied in the range from 1/60 to 1/15. The optimal parameters for the DEMM were found to be a learning rate of 1e-3, a hidden dimension of 30, and a λ of 1/30. The CNN has 4 layers and the local FCN has 3 layers. All deep learning models were trained for 50 epochs and early stopping was used to save model parameters that achieved the lowest validation loss during training.

All prediction windows were randomly assigned to the train, validation, or test set. A total of 450 prediction windows were assigned to the training set, 250 to the validation set, and the remainder to the test set. A total of 10 random train-validation-test splits were used.

Below we consider the following evaluation metrics:

- **NLL** - The average negative log likelihood of test samples given each model's predicted conditional distribution.

- **MSE** - The mean squared error of each model's point prediction.

- **F1 Micro** - Test precipitation samples are assigned to 1 of 3 classes: zero rainfall, non-zero non-extreme rainfall, and extreme rainfall where extreme rainfall is rainfall that exceeds U_{lw}.

For each model, samples can be assigned class probabilities using that model's CDF (e.g. a sample's probability of belonging to the excess class will be 1 minus the model's CDF evaluated at the threshold) and then assigned to the class with the highest probability. Note that in the case of the *DEMM* the class probabilities will be the same as mixture weights. The predicted and ground truth class labels can then be used to compute F1 micro.

- **F1 Macro** - As in the case of F1 micro, each test sample can be assigned a predicted class for each model and a ground truth class. These can then be used to compute F1 macro.

6.5.4 Experimental Results

This section presents the results of our experiments. In addition to characterizing the overall performance of *DEMM*, the experiments were designed to:

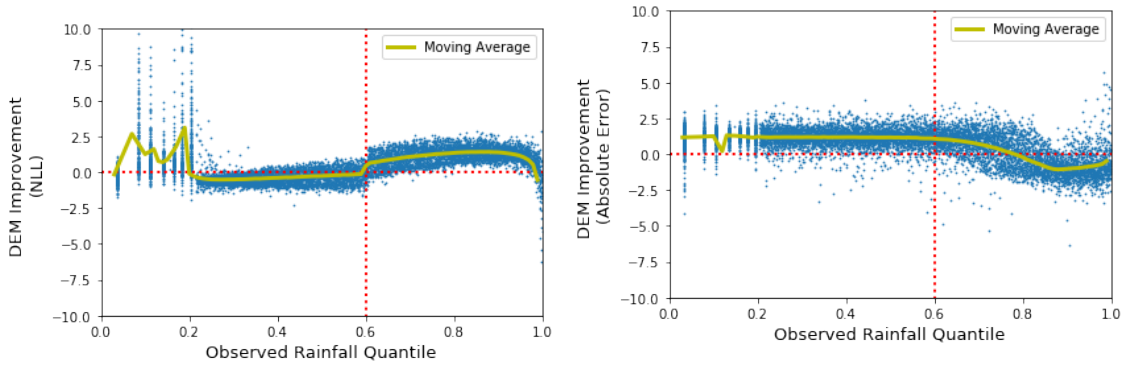
1. Perform an ablation study to compare the performance of *DEMM* when trained using a fixed versus variable threshold defined for extreme values.
2. Characterize the spatial locations where the *DEMM* outperforms the strongest baseline.
3. Evaluate the ability of the *DEMM* to predict the frequency of extreme events.
4. Evaluate the ability of the *DEMM* to predict the timing of extreme events.

6.5.4.1 Performance Comparison against Baseline Methods

Table 6.1 compares the predictive performance of *DEMM* against the baseline methods. For evaluation purposes, the excess threshold U_{l_w} was set to the global 0.6 quantile value. We consider two variations of *DEMM*, one with a fixed excess threshold, also at 0.6 quantile, while the other was trained with varying thresholds as described in Section 6.4.4. In the latter case, the results are reported when the threshold is set to the 0.6 quantile at test time. Our experimental results show that both versions of *DEMM* (fixed and variable threshold) outperform the baseline methods in terms of their negative log likelihood, MSE, and F1 score. The *DEMM* achieves lower MSE than the

	NLL	MSE	F1 Micro	F1 Macro
Ensemble Mean	N/A	15.48 ± 0.56	N/A	N/A
Kernel Density Estimation	2.787 ± 0.083	15.48 ± 0.56	0.500 ± 0.006	0.323 ± 0.002
Lognormal Hurdle Model	0.906 ± 0.076	16.39 ± 0.51	0.543 ± 0.011	0.258 ± 0.020
DEMM (fixed threshold)	0.412 ± 0.074	14.94 ± 0.55	**0.612** ± 0.007	**0.416** ± 0.005
DEMM (variable threshold)	**0.411** ± 0.080	**14.70** ± 0.59	0.610 ± 0.007	0.413 ± 0.008

Table 6.1: Predictive performance comparison



(a) Improvement over the baseline in NLL

(b) Improvement over the baseline in absolute error.

Figure 6.2: Plots showing the improvement of the *DEMM* over a hurdle model as the amount of rainfall varies. Positive values indicate samples where the *DEMM* has achieves a lower loss than hurdle model baseline.

ensemble mean. Because the ensemble mean is expected to be a strong baseline, this demonstrates that the *DEMM*'s ability to make accurate point predictions was not strongly inhibited by simultaneously predicting the conditional distribution. Interestingly, the hurdle baseline performs worse than the ensemble mean, which demonstrates the importance of correctly specifying the probability distribution of the data. The fact that *DEMM* outperforms the hurdle model demonstrates the value of incorporating extreme value theory into the mixture model.

A more detailed comparison between the relative performance of *DEMM* with variable threshold against the hurdle baseline model is shown in Figure 6.2. The quantile values of the observed rainfall are plotted along the x-axis and the difference between the losses of the two models is plotted along the y-axis. Positive values indicate that the *DEMM* is outperforming the baseline for a given sample. We find that for NLL the *DEMM* model achieves its best performance relative to the baseline for samples that are excesses above the threshold. In other words, the *DEMM* outperforms the baseline

in predicting the occurence of excess values. However, in the case of MSE, the *DEMM* outperforms the baseline above the threshold less consistently.

6.5.4.2 Ablation Studies

Since the *DEMM* was trained to accommodate variable thresholds, it is important to consider what performance penalty this imposes over training with a single fixed threshold. Surprisingly, we find that the performance of the *DEMM* remains virtually unchanged when trained with a fixed or variable threshold. Table 6.1 shows that regardless of the chosen metrics, the two methods results in almost identical performance. In Figure 6.3, we show that the *DEMM* (with variable threshold) performs similarly to its fixed threshold variant for a range of thresholds. This encouraging result suggests that training the model with a variable threshold imposes little if any performance penalties on the model while providing additional flexibility at test time.

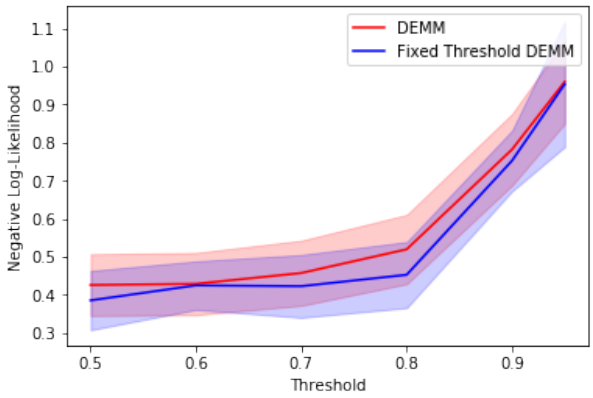
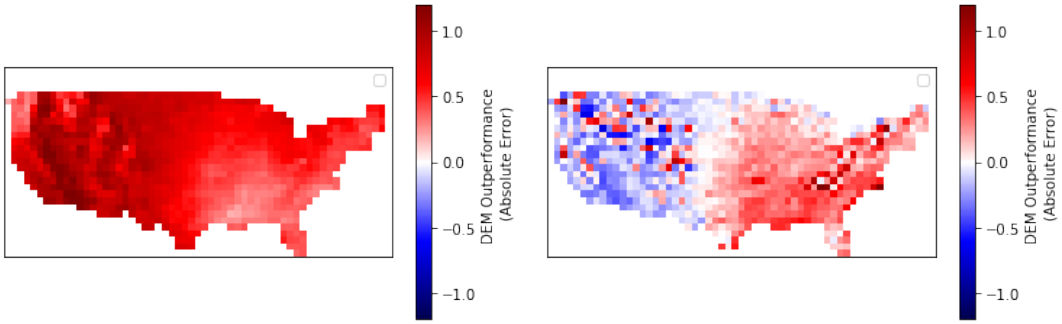


Figure 6.3: Plot showing how the negative log-likelihood varies for both the *DEMM* and the fixed threshold variant of the *DEMM*. Shaded region represents ±1 standard deviation.

6.5.4.3 Spatial Analysis

Figure 6.4 shows the spatial relationships between locations where the variable-threshold *DEMM* improves over baselines. These figures were created by averaging the mean absolute error (MAE) achieved at each location by each of the respective models (*DEMM*, hurdle, or ensemble mean) across all samples and data splits. The average MAE of the *DEMM* at each of location is subtracted



(a) Improvement over hurdle model. (b) Improvement over the ensemble mean.

Figure 6.4: Plots showing the spatial locations where the *DEMM* outperforms the hurdle model in terms of their mean absolute error (MAE). Red indicates that the *DEMM* is outperforming.

from the corresponding MAE for one of the baselines so that positive values shaded in red represent locations where *DEMM* outperforms one of the baselines. Based on these figures, it is easy to see that the *DEMM* outperforms the hurdle model at almost all locations. However, consistent with the results given in Table 6.1, we see that the improvement in absolute error over the ensemble mean baseline is less consistent. Nevertheless, the improvements over the ensemble mean baseline are concentrated at the locations where precipitation values are largest and most variable. This can be verified with Figure 6.5, where we plot the average rainfall at each location on the x-axis and the margin by which the *DEMM* outperforms the ensemble mean baseline in terms of absolute error on the y-axis. The strong correlation (0.58) between the two variables indicates that the higher average rainfall at a location is, the better the *DEMM* performs relative to the ensemble mean.

6.5.4.4 Extreme Event Frequency

Besides the overall prediction error, another important metric for evaluating the performance of the different methods is their ability to correctly predict the frequency of extreme events. Because different applications may be interested in defining extreme values differently, in Figure 6.7 we plot predicted vs empirical frequency of extreme values at varying thresholds. The empirical threshold is the quantile defining the excess threshold while the predicted frequency is calculated by computing the probability of an excess value occurring (as described in Section 6.5.3) averaged

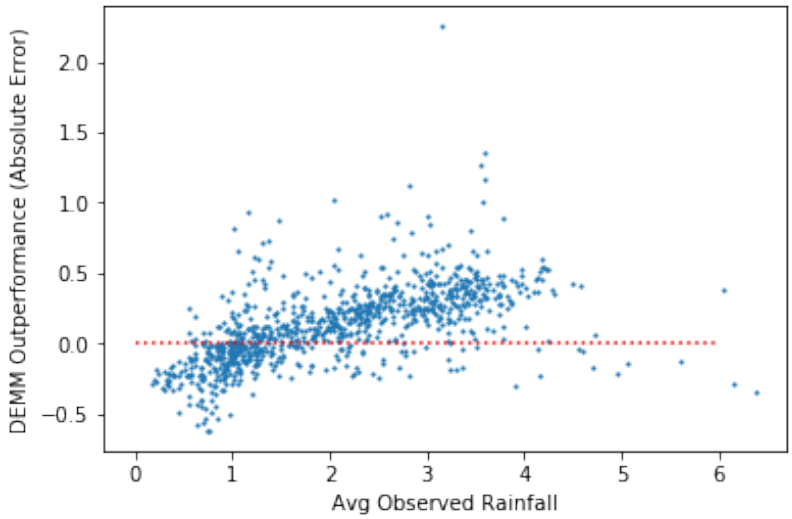


Figure 6.5: The relationship between average precipitation at a location and the margin by which the *DEMM* outperforms the ensemble mean at that location. Positive values indicate the *DEMM* outperforms.

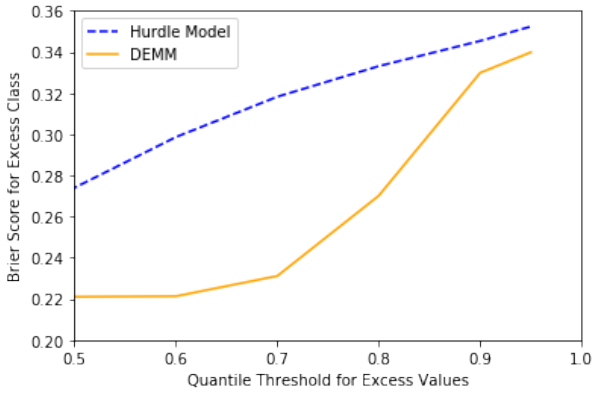


Figure 6.6: Plot showing how the brier score for the extreme events class varies with the quantile used to define extreme events.

across all samples in the test set. We find that the hurdle model consistently under predicts the frequency of extreme values regardless of the quantile threshold used while the *DEMM* accurately predicts their frequency across all thresholds. This suggests that the *DEMM* is well suited for predicting the frequency of extreme events regardless of the threshold used to define them.

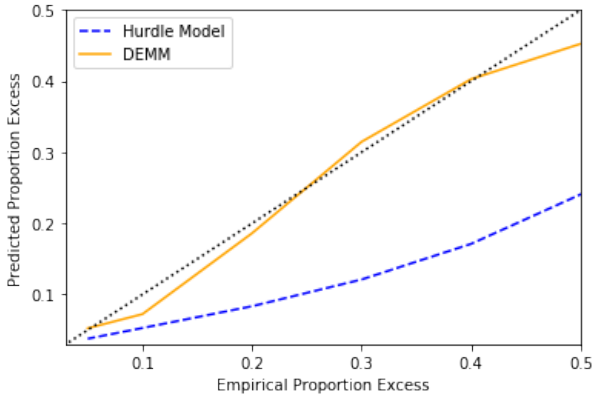


Figure 6.7: Plot comparing the observed frequency of extreme events (x-axis) against the predicted frequency of extreme events.

6.5.4.5 Extreme Event Timing

In addition to predicting their frequencies. it is also important to predict their precise timing of the extreme events. To evaluate this we compare the Brier score of the *DEMM* and hurdle model using a variety of thresholds to define extreme values in Figure 6.6. The Brier score is a classification metric with a lower Brier score representing better predictive performance. Given the set of binary class labels for every observed precipitation value, $\{Y_i | i \in \{1, \cdots, n\}\}$, which represents whether or not each sample is an extreme value, and the set of predicted probability of excess for each sample, $\{\hat{Y}_i | i \in \{1, \cdots, n\}\}$, then the Brier score can be computed as:

$$\sum_{i=1}^{n} \frac{(Y_i - \hat{Y}_i)^2}{n} \tag{6.16}$$

The results shown in the figure suggest that the *DEMM* consistently outperforms the hurdle model regardless of the threshold chosen to define extreme events. Therefore the *DEMM* is better at predicting when excess precipitation values will occur.

6.6 Conclusion

This chapter introduces a new deep learning framework called *DEMM* for predicting geo-spatio-temporal events. The proposed framework centers around a mixture model that incorporates extreme value theory to accurately model the distribution of extreme events while simultaneously making accurate point predictions. The framework employs a set of novel re-parameterization techniques

to ensure that the neural network outputs satisfy the constraints placed on the parameters of the mixture model, including a constraint on the GPD shape parameter required for computing the mean of the mixture model. Furthermore, the proposed framework also allows the excess threshold to be an input to the model, thus providing flexibility for the user to alter the threshold at test time without the need to retrain the model. The experiments performed on a real world precipitation data set showed that *DEMM* is able to accurately predict the timing and frequency of extreme events while also making accurate point predictions and distribution predictions. The experimental results also verified that the proposed technique for allowing the thresholds to modified without retraining the model imposes no performance penalty.

CHAPTER 7

CONCLUSION AND FUTURE WORK

This dissertation has demonstrated how deep learning can be effectively applied to geo-spatio-temporal prediction tasks. In particular, novel deep learning frameworks have been proposed to address the various challenging aspects of geo-spatio-temporal relationships, including non-linearity, temporal relationships, and spatial relationships in both predictors and model parameters. Because extreme events are of particular interest in many geo-spatio-temporal applications, considerable attention has been devoted to the accurate modeling of extreme events, including their magnitude, timing, frequency, and overall distribution.

In Chapter 3, a deep learning architecture known as WGC-LSTM was developed for geo-spatio-temporal prediction. The framework incorporated graph convolution into an LSTM network to model spatial relationships among irregularly spaced locations and to infer the relatedness of the locations in a data driven way. Experimental results not only demonstrated the improved accuracy of the proposed approach but also showed that WGC-LSTM could yield interpretable insights into the spatial patterns underlying the data.

In Chapter 4, we demonstrated that graph convolution could be used not just to model spatial relationships among predictors but among model parameters as well. Furthermore, we provided some suggestions for when convolution should be used to model spatial relationships among model parameters and when it should be used to model spatial relationships among predictors. We proposed a graph convolution based architecture capable of simultaneously modeling spatial relationships among predictors and model parameters. Our experiments demonstrated the strong predictive capabilities of our proposed approach as well as validated our advice regarding when to apply convolution for spatial modeling.

Chapter 5 focuses on modeling the distribution of extreme events. We showed how a re-parameterization trick could be used to allow a neural network to predict the parameters of a generalized Pareto distribution while ensuring all the necessary constraints are satisfied. In addition,

we explained how deep sets could be used to combine set valued predictors and vector valued predictors. We demonstrated the effectiveness of our approach on a real world precipitation data set.

Finally, in Chapter 6 we extended the results in Chapter 5 to allow users to simultaneously predict the distribution of extreme events while also making point predictions. Accomplishing this required us to model the full conditional distribution and also required us to further constrain the generalized Pareto distribution parameters to ensure that its expected value is well-defined. A crucial aspect of the proposed *DEMM* framework was the choice of threshold used to define excess values. We designed our architecture to allow the threshold to be altered without needing to retrain the model. Using a real world precipitation data set, we were able to show that *DEMM* could accurately predict the magnitude, timing, frequency, and overall distribution of extreme events.

7.1 Future Work

The application of deep learning to spatio-temporal data presents many additional avenues for further research. We will focus on two broad areas of potential future research. First we will discuss new areas of research in modeling spatio-temporal relationships and second, further research opportunities in the prediction of extreme spatio-temporal events.

7.1.1 Spatio-temporal Relationships

One fundamental problem in applying deep learning to spatio-temporal data is how to effectively incorporate such relationships into the model formulation. So far, existing works have made heavy use of Tobler's law, which states that things that are closer are more related than those that are further apart. However, some phenomena that buck this trend. In climate, for instance, long distance relationships called teleconnections have been well-studied and are known to be related to important phenomena like El Nino. Convolution is primarily used for modeling relationships in spatially localized area. Therefore, new architectures drawing on techniques such as attention will likely be required for inferring these long distance dependencies. One promising approach for

modeling these relationships is with transformers [96]. Transformers have recently seen a wide number of applications including to images [28] and sequence data [96]. Though transformers have strong capabilities in modelling relationships between distant parts of an input, their computational requirement grows quickly with the size of the input. In cases where we would like to find distant relationships in high resolution global scale data or in spatio-temporal data with long sequence lengths, naive application of transformers becomes computationally infeasible. Developing techniques for applying transformers at these scales is thus an important future research problem.

7.1.2 Extreme Events

Thus far, our study of deep learning and extreme value theory has focused on the generalized Pareto distribution which models the distribution of excess values above a threshold. However, another distribution fundamental to the study of extreme values is the generalized extreme value (GEV) distribution which governs the distribution of block maxima. Similar to the generalized Pareto distribution, the GEV is backed by a robust body of theoretical results, justifying its use as a technique for modeling block maxima in a wide range of circumstances.The GEV distribution also has constraints placed on its parameters, so any deep learning approach that attempts to infer these parameter values must find ways to enforce the constraints. The re-parameterization approach applied to the GPD will need to be adapted to accomplish this. In addition, the use of the GEV distribution rather than the GPD will significantly reduce the amount of data available for training since, within each prediction window, it is possible to have multiple samples of excess values from the GPD but only a single block maxima value to fit the GEV distribution. In applications where data is limited this could pose a significant challenge as deep learning based approaches often benefit from having large amounts of training data available.

A second problem is how to develop deep learning approaches for modeling the joint distribution of multivariate extremes. Most of current research has focused on modeling the spatio-temporal relationships among predictors or model parameters, but there may be cases where we would like to know how the multivariate extreme events observed across different times and locations will

vary spatio-temporally. For instance, it would be useful to know the duration of extreme rainfall at a given location or the spatial extent of flooding. Although there exists some previous work that investigates the distribution of multivariate extremes using generalizations of the GEV and GPD or copula theory, there is comparatively little work integrating these approaches with deep learning.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Abraham, Z.; Liszewska, M.; Perdinan; Tan, P.-N.; Winkler, J.; and Zhong, S. 2013. Distribution regularized regression framework for climate modeling. In *Proceedings of SIAM International Conference on Data Mining*, 333–341.

[2] Albert, A.; Strano, E.; Kaur, J.; and González, M. 2018. Modeling Urbanization Patterns with Generative Adversarial Networks. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2095–2098. ISSN: 2153-6996.

[3] Baccouche, M.; Mamalet, F.; Wolf, C.; Garcia, C.; and Baskurt, A. 2011. Sequential Deep Learning for Human Action Recognition. In Hutchison, D.; Kanade, T.; Kittler, J.; Kleinberg, J. M.; Mattern, F.; Mitchell, J. C.; Naor, M.; Nierstrasz, O.; Pandu Rangan, C.; Steffen, B.; Sudan, M.; Terzopoulos, D.; Tygar, D.; Vardi, M. Y.; Weikum, G.; Salah, A. A.; and Lepri, B., eds., *Human Behavior Unterstanding*, volume 7065. Berlin, Heidelberg: Springer Berlin Heidelberg. 29–39.

[4] Bacry, E.; Gaïffas, S.; Kabeshova, A.; and Yu, Y. 2019. Zimm: a deep learning model for long term adverse events with non-clinical claims data. *arXiv preprint arXiv:1911.05346*.

[5] Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv:1803.01271 [cs]*. arXiv: 1803.01271.

[6] Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gulcehre, C.; Song, F.; Ballard, A.; Gilmer, J.; Dahl, G.; Vaswani, A.; Allen, K.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; and Pascanu, R. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261 [cs, stat]*. arXiv: 1806.01261.

[7] Bauer, P.; Thorpe, A.; and Brunet, G. 2015. The quiet revolution of numerical weather prediction. *Nature* 525(7567):47–55.

[8] Bergstra, J., and Bengio, Y. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research* 13(Feb):281–305.

[9] Bishop, C. M. 1994. Mixture density networks.

[10] Boriah, S.; Kumar, V.; Steinbach, M.; Tan, P.; Potter, C.; and Klooster, S. 2008. Detecting ecosystem disturbances and land cover change using data mining. In Kargupta, H.; Han, J.; and Yu, P. S., eds., *Next Generation of Data Mining*. Chapman & Hall/CRC.

[11] Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric

deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine* 34(4):18–42. arXiv: 1611.08097.

[12] Bruna, J.; Zaremba, W.; Szlam, A.; and Lecun, Y. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR)*.

[13] Canfield, R. V.; Olsen, D.; Hawkins, R.; and Chen, T. 1980. Use of extreme value theory in estimating flood peaks from mixed populations.

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

[15] Castro, R.; Souto, Y. M.; Ogasawara, E.; Porto, F.; and Bezerra, E. 2021. Stconvs2s: Spatiotemporal convolutional sequence to sequence network for weather forecasting. *Neurocomputing* 426:285–298.

[16] Chen, B.; Ting, J.-A.; Marlin, B.; and de Freitas, N. Deep Learning of Invariant Spatio-Temporal Features from Video. 9.

[17] Cheng, H.; Tan, P.; Potter, C.; and Klooster, S. 2008a. A robust graph-based algorithm for detection and characterization of anomalies in noisy multivariate time series. In *Proceedings of ICDM Workshop on Spatial and Spatio-temporal Data Mining (STDM 08)*.

[18] Cheng, H.; Tan, P.-N.; Potter, C.; and Klooster, S. 2008b. Data mining for visual exploration and detection of ecosystem disturbaces. In *Proc of 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*.

[19] Cheng, H.; Tan, P.-N.; Potter, C.; and Klooster, S. 2009. Detection and characterization of anomalies in multivariate time series. In *Proceedings of SIAM International Conference on Data Mining*.

[20] Cho, K.; van Merrienboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 103–111. Doha, Qatar: Association for Computational Linguistics.

[21] Coles, S.; Bawa, J.; Trenner, L.; and Dorazio, P. 2001. *An introduction to statistical modeling of extreme values*, volume 208. Springer.

[22] Coles, S. 2001. *An Introduction to Statistical Modeling of Extreme Values*. Springer.

[23] Collins, S.; Yuan, S.; Tan, P.; Oliver, S.; Lapierre, J.; Cheruvelil, K.; Fergus, C.; Skaff, N.; Stachelek, J.; Wagner, T.; et al. 2019. Winter precipitation and summer temperature predict lake water quality at macroscales. *Water Resources Research*.

[24] Cragg, J. G. 1971. Some statistical models for limited dependent variables with application

to the demand for durable goods. *Econometrica: Journal of the Econometric Society* 829–844.

[25] Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 3844–3852.

[26] Ding, D.; Zhang, M.; Pan, X.; Yang, M.; and He, X. 2019. Modeling Extreme Events in Time Series Prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19*, 1114–1122. Anchorage, AK, USA: ACM Press.

[27] Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2625–2634.

[28] Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[29] Gagne, D. J.; Haupt, S. E.; Nychka, D. W.; and Thompson, G. 2019. Interpretable deep learning for spatial analysis of severe hailstorms. *Monthly Weather Review* (2019).

[30] Gavin, J. 2000. Extreme value theory-an empirical analysis of equity risk.

[31] Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. *arXiv:1704.01212 [cs]*. arXiv: 1704.01212.

[32] Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580–587. Columbus, OH, USA: IEEE.

[33] Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press.

[34] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

[35] Graves, A.; Jaitly, N.; and Mohamed, A. r. 2013. Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 273–278.

[36] Grover, A.; Kapoor, A.; and Horvitz, E. 2015. A Deep Hybrid Model for Weather Forecasting.

379–386. ACM Press.

[37] Hamilton, W. L.; Ying, R.; and Leskovec, J. 2018. Inductive Representation Learning on Large Graphs. *arXiv:1706.02216 [cs, stat]*. arXiv: 1706.02216.

[38] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. 770–778.

[39] Hirose, H., and Wang, L. 2012. Prediction of infectious disease spread using twitter: A case of influenza. In *Fifth Int'l Symp. on Parallel Architectures, Algorithms and Programming*, 100–105. IEEE.

[40] Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

[41] Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5):359–366.

[42] Hossain, M.; Rekabdar, B.; Louis, S. J.; and Dascalu, S. 2015. Forecasting the weather of Nevada: A deep learning approach. In *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–6.

[43] Ji, S.; Xu, W.; Yang, M.; and Yu, K. 2010. 3d Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35:221–231.

[44] Ji, S.; Xu, W.; Yang, M.; and Yu, K. 2012. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence* 35(1):221–231.

[45] Jozefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; and Wu, Y. 2016. Exploring the Limits of Language Modeling. *arXiv:1602.02410 [cs]*. arXiv: 1602.02410.

[46] Kaiser, P.; Wegner, J. D.; Lucchi, A.; Jaggi, M.; Hofmann, T.; and Schindler, K. 2017. Learning Aerial Image Segmentation From Online Maps. *IEEE Transactions on Geoscience and Remote Sensing* 55(11):6054–6068.

[47] Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 1725–1732.

[48] Katz, R. W.; Parlange, M. B.; and Naveau, P. 2002. Statistics of extremes in hydrology. *Advances in water resources* 25(8-12):1287–1304.

[49] Kharin, V. V., and Zwiers, F. W. 2005. Estimating extremes in transient climate change simulations. *Journal of Climate* 18(8):1156–1173.

[50] Kingma, D., and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

[51] Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

[52] Kong, S.; Bai, J.; Lee, J. H.; Chen, D.; Allyn, A.; Stuart, M.; Pinsky, M.; Mills, K.; and Gomes, C. P. 2020. Deep hurdle networks for zero-inflated multi-target regression: Application to multiple species abundance estimation. *arXiv preprint arXiv:2010.16040*.

[53] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F.; Burges, C. J. C.; Bottou, L.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc. 1097–1105.

[54] Kuai, F.; Chaopeng, S.; Daniel, K.; and Xiao, Y. Prolongation of smap to spatiotemporally seamless coverage of continental u.s. using a deep learning neural network. *Geophysical Research Letters* 44(21):11,030–11,039.

[55] Kuligowski, R. J., and Barros, A. P. 1998. Localized Precipitation Forecasts from a Numerical Weather Prediction Model Using Artificial Neural Networks. *Weather and Forecasting* 13(4):1194–1204.

[56] Laptev, N.; Yosinski, J.; Li, L. E.; and Smyl, S. 2017. Time-series extreme event forecasting with neural networks at uber. In *International Conference on Machine Learning*, volume 34, 1–5.

[57] Lebedev, V.; Ivashkin, V.; Rudenko, I.; Ganshin, A.; Molchanov, A.; Ovcharenko, S.; Grokhovetskiy, R.; Bushmarinov, I.; and Solomentsev, D. 2019. Precipitation Nowcasting with Satellite Imagery. *arXiv:1905.09932 [cs]*. arXiv: 1905.09932.

[58] LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553):436–444.

[59] LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation* 1(4):541–551.

[60] LeCun, Y.; Boser, B. E.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. E.; and Jackel, L. D. 1990. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, 396–404.

[61] Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR '18)*.

[62] Li, J.; Han, Z.; Cheng, H.; Su, J.; Wang, P.; Zhang, J.; and Pan, L. 2019. Predicting Path Failure In Time-Evolving Graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19*, 1279–1289. Anchorage, AK, USA: ACM Press.

[63] Liu, J. N.; Hu, Y.; You, J. J.; and Chan, P. W. 2014. Deep neural network based feature representation for weather forecasting. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[64] Liu, X.; Tan, P.-N.; Abraham, Z.; Luo, L.; and Hatami, P. 2018a. Distribution preserving multi-task regression for spatio-temporal data. In *Proc of ICDM*, 1134–1139. IEEE.

[65] Liu, X.; Tan, P.-N.; Abraham, Z.; Luo, L.; and Hatami, P. 2018b. Distribution preserving multi-task regression for spatio-temporal data. In *Proceedings of IEEE International Conference on Data Mining*.

[66] Liu, X.; Wilson, T.; Tan, P.-N.; and Luo, L. 2019. Hierarchical lstm framework for long-term sea surface temperature forecasting. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 41–50. IEEE.

[67] López, J., and Francés, F. 2013. Non-stationary flood frequency analysis in continental spanish rivers, using climate and reservoir indices as external covariates. *Hydrology and Earth System Sciences* 17(8):3189–3203.

[68] Mattyus, G.; Luo, W.; and Urtasun, R. 2017. DeepRoadMapper: Extracting Road Topology From Aerial Images. 3438–3446.

[69] McGovern, A.; Hiers, N. C.; Collier, M.; Gagne II, D. J.; and Brown, R. A. 2008. Spatiotemporal Relational Probability Trees: An Introduction. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, 935–940. Washington, DC, USA: IEEE Computer Society.

[70] Nasrollahi, N.; Hsu, K.; and Sorooshian, S. 2013. An Artificial Neural Network Model to Reduce False Alarms in Satellite Precipitation Products Using MODIS and CloudSat Observations. *Journal of Hydrometeorology* 14(6):1872–1883.

[71] Nayak, M. A., and Ghosh, S. 2013. Prediction of extreme rainfall event using weather pattern recognition and support vector machine classifier. *Theoretical and applied climatology* 114(3-4):583–603.

[72] Oord, A. v. d.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; and Kavukcuoglu, K. 2016. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499 [cs]*. arXiv: 1609.03499.

[73] Pan, Z.; Liang, Y.; Wang, W.; Yu, Y.; Zheng, Y.; and Zhang, J. 2019. Urban Traffic Prediction from Spatio-Temporal Data Using Deep Meta Learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19*, 1720–1730. Anchorage, AK, USA: ACM Press.

[74] Racah, E.; Beckham, C.; Maharaj, T.; Kahou, S. E.; Pal, C.; et al. 2016. Extremeweather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. *arXiv preprint arXiv:1612.02095*.

[75] Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–1144.

[76] Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N.; Hornegger, J.; Wells, W. M.; and Frangi, A. F., eds., *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, 234–241. Cham: Springer International Publishing.

[77] Sadeghi, M.; Asanjan, A. A.; Faridzad, M.; Nguyen, P.; Hsu, K.; Sorooshian, S.; and Braithwaite, D. 2019. Persiann-cnn: Precipitation estimation from remotely sensed information using artificial neural networks–convolutional neural networks. *Journal of Hydrometeorology* 20(12):2273–2289.

[78] Scarf, P. A., and Laycock, P. 1994. Applications of extreme value theory in corrosion engineering. *JOURNAL OF RESEARCH-NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY* 99:313–313.

[79] Shen, L.; Mickley, L. J.; and Gilleland, E. 2016. Impact of increasing heat waves on us ozone episodes in the 2050s: Results from a multimodel analysis using extreme value theory. *Geophysical research letters* 43(8):4017–4025.

[80] Shi, X.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.-k.; and Woo, W.-c. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *NeurIPS 28*. 802–810.

[81] Simonyan, K., and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*. arXiv: 1409.1556.

[82] Sun, T.; Chen, Z.; Yang, W.; and Wang, Y. 2018. Stacked U-Nets with Multi-output for Road Extraction. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 187–1874. Salt Lake City, UT, USA: IEEE.

[83] Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

[84] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going Deeper With Convolutions. 1–9.

[85] Tao, Y.; Gao, X.; Hsu, K.; Sorooshian, S.; and Ihler, A. 2016a. A Deep Neural Network Modeling Framework to Reduce Bias in Satellite Precipitation Products. *Journal of Hydrometeorology* 17(3):931–945.

[86] Tao, Y.; Gao, X.; Ihler, A.; Hsu, K.; and Sorooshian, S. 2016b. Deep neural networks for precipitation estimation from remotely sensed information. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, 1349–1355. ISSN: null.

[87] Tao, Y.; Gao, X.; Ihler, A.; Sorooshian, S.; and Hsu, K. 2017. Precipitation Identification with Bispectral Satellite Information Using Deep Learning Approaches. *Journal of Hydrometeorology* 18(5):1271–1283.

[88] Tobler, W. 1970. A computer movie simulating urban growth in the detroit region. *Economic Geography* 46(2):234–240.

[89] Toshev, A., and Szegedy, C. 2014. DeepPose: Human Pose Estimation via Deep Neural Networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 1653–1660. Columbus, OH, USA: IEEE.

[90] Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M. 2015a. Learning Spatiotemporal Features with 3d Convolutional Networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 4489–4497. Santiago, Chile: IEEE.

[91] Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M. 2015b. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, 4489–4497.

[92] Tran, D.; Ray, J.; Shou, Z.; Chang, S.-F.; and Paluri, M. 2017. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*.

[93] Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; and Paluri, M. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6450–6459. Salt Lake City, UT: IEEE.

[94] Vandal, T.; Kodra, E.; Ganguly, S.; Michaelis, A.; Nemani, R.; and Ganguly, A. R. 2017. DeepSD: Generating High Resolution Climate Change Projections Through Single Image Super-Resolution. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, 1663–1672. New York, NY, USA: ACM. event-place: Halifax, NS, Canada.

[95] Vandal, T.; Kodra, E.; Dy, J.; Ganguly, S.; Nemani, R.; and Ganguly, A. R. 2018. Quantifying uncertainty in discrete-continuous and skewed data with bayesian deep learning. In *Proceedings*

of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2377–2386.

[96] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

[97] Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. *arXiv:1710.10903 [cs, stat]*. arXiv: 1710.10903.

[98] Viroli, C., and McLachlan, G. J. 2019. Deep gaussian mixture models. *Statistics and Computing* 29(1):43–51.

[99] Waibel, A.; Hanazawa, T.; Hinton, G.; Shikano, K.; and Lang, K. 1989. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37(3):328–339.

[100] Weng, T.-W.; Zhang, H.; Chen, P.-Y.; Yi, J.; Su, D.; Gao, Y.; Hsieh, C.-J.; and Daniel, L. 2018. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*.

[101] Wilson, T.; Tan, P.-N.; and Luo, L. 2018. A Low Rank Weighted Graph Convolutional Approach to Weather Prediction. *Proc of ICDM* 627–636.

[102] Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32(1):4–24.

[103] Wu, M. Z.; Luo, J.; Fang, X.; Xu, M.; and Zhao, P. 2021. Modeling multivariate cyber risks: Deep learning dating extreme value theory. *arXiv preprint arXiv:2103.08450*.

[104] Xu, J.; Tan, P.-N.; Luo, L.; and Zhou, J. 2016. Gspartan: a geospatio-temporal multi-task learning framework for multi-location prediction. In *Proceedings of SIAM International Conference on Data Mining*.

[105] Xu, J.; Tan, P.-N.; Zhou, J.; and Luo, L. 2017. Online multi-task learning framework for ensemble forecasting. *IEEE Transactions on Knowledge and Data Engineering* 29(6):1268–1280.

[106] Ye, J.; Sun, L.; Du, B.; Fu, Y.; Tong, X.; and Xiong, H. 2019. Co-Prediction of Multiple Transportation Demands Based on Deep Spatio-Temporal Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19*, 305–313. Anchorage, AK, USA: ACM Press.

[107] You, J.; Li, X.; Low, M.; Lobell, D.; and Ermon, S. 2017. Deep gaussian process for crop

yield prediction based on remote sensing data. In *AAAI*.

[108] Yu, R.; Li, Y.; Shahabi, C.; Demiryurek, U.; and Liu, Y. 2017. Deep Learning: A Generic Approach for Extreme Condition Traffic Forecasting. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, Proceedings. Society for Industrial and Applied Mathematics. 777–785.

[109] Yu, X.; Zhao, Z.; Zhang, X.; Zhang, Q.; Liu, Y.; Sun, C.; and Chen, X. 2021. Deep learning-based open set fault diagnosis by extreme value theory. *IEEE Transactions on Industrial Informatics*.

[110] Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* 3634–3640. arXiv: 1709.04875.

[111] Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; and Smola, A. J. 2017. Deep sets. In *Advances in neural information processing systems*, 3391–3401.

[112] Zaytar, M. A., and Amrani, C. E. 2016. Sequence to Sequence Weather Forecasting with Long Short-Term Memory Recurrent Neural Networks. *International Journal of Computer Applications* 143(11):7–11.

[113] Zhang, Z.; Liu, Q.; and Wang, Y. 2018. Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters* 15(5):749–753.

[114] Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2019. Graph Neural Networks: A Review of Methods and Applications. *arXiv:1812.08434 [cs, stat]*. arXiv: 1812.08434.

[115] Zhou, J.; Chen, J.; and Ye, J. 2011. Malsar: Multi-task learning via structural regularization. *Arizona State University*.

[116] Zhou, D.-X. 2020. Universality of deep convolutional neural networks. *Applied and computational harmonic analysis* 48(2):787–794.

[117] Zong, B.; Song, Q.; Min, M. R.; Cheng, W.; Lumezanu, C.; Cho, D.; and Chen, H. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*.