EXTRACTING ABSTRACT SPATIO-TEMPORAL FEATURES OF WEATHER PHENOMENA FOR AUTOENCODER TRANSFER LEARNING

by

Richard Arthur McAllister

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

April, 2020

DEDICATION

For Neil Ellwood Peart - a pioneer in this field.
May “The Scientist” rest in peace!

Each of us...a cell of awareness
imperfect and incomplete.
Genetic blends, with uncertain ends
on a fortune hunt that’s far too fleet.

ACKNOWLEDGEMENTS

Dr. John Sheppard: My advisor and committee chair. ...an excellent man. He pushed me, and I needed it. In setting up, inspiring, mentoring, and managing the Numerical Intelligent Systems Laboratory, he has accomplished a great academic feat. In getting me to finish this dissertation, he has accomplished the impossible. In the vein of all those who ever said, “you’ll thank me later” he has certainly distinguished himself. Thank you, John. **Dr. Clemente Izurieta, Dr. David Millman, and Dr. John Sample:** These are my committee members. Thank you all for your advice and support! **Dr. Maryann Cummings:** You stepped in at a most critical time. Thank you! **The fine students and graduates of Montana State University’s Numerical Intelligent Systems Laboratory** This is the fantastic group of Dr. Sheppard’s students who contribute excellence to the Machine Learning community. May you all continue to immerse yourselves in this subject matter with zeal, alacrity, aplomb, and good humor! **Dr. Albin Gasiewski and Dr. Kun Zhang** They are both from the University of Colorado. They provided help with the data generation from the WRF system. Thank you! In alphabetical order, **Dave Gallaher, Noel Heustis, Pete Heidmann, Bill Hosack, and Jenny Lane**...you inspired me, listened to me, laughed with me, and enabled my prolonged adolescence. Your endurance is prodigious. You are better people than I, and I am honored to call you my friends. **My brother, Francis:** You got me into this mess in the first place! **The rest of my family:** My family is simply enormous. Thank you all for putting up with me through all of this! **My parents, Frank and Marcia McAllister:** Your patience and support was unreasonable, unwise, unwarranted, and stubborn. Thank you far more than anyone on this or the previous page! I love you and I owe it to you to become fantastic!

TABLE OF CONTENTS

TABLE OF CONTENTS - CONTINUED

TABLE OF CONTENTS - CONTINUED

TABLE OF CONTENTS - CONTINUED

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES - CONTINUED

LIST OF FIGURES - CONTINUED

LIST OF FIGURES - CONTINUED

LIST OF FIGURES - CONTINUED

LIST OF FIGURES - CONTINUED

LIST OF FIGURES - CONTINUED

LIST OF ALGORITHMS

ABSTRACT

In this dissertation we develop ways to discover encodings within autoencoders that can be used to exchange information among neural network models.

We begin by verifying that autoencoders can be used to make predictions in the meteorological domain, specifically for wind vector determination. We use unsupervised pre-training of stacked autoencoders to construct multilayer perceptrons to accomplish this task. We then discuss the role of our approach as an important step in positioning Empirical Weather Prediction as a viable alternative to Numerical Weather Prediction.

We continue by exploring the spatial extensibility of the previously developed models, observing that different areas in the atmosphere may be influenced unique forces. We use stacked autoencoders to generalize across an area of the atmosphere, expanding the application of networks trained in one area to the surrounding areas. As a prelude to exploring transfer learning, we demonstrate that a stacked autoencoder is capable of capturing knowledge universal to these dataspaces.

Following this we observe that in extremely large dataspaces, a single neural network covering that space may not be effective, and generating large numbers of deep neural networks is not feasible. Using functional data analysis and spatial statistics we analyze deep networks trained from stacked autoencoders in a spatio-temporal application area to determine the extent to which knowledge can be transferred to similar regions. Our results indicate high likelihood that spatial correlation can be exploited if it can be identified prior to training.

We then observe that artificial neural networks, being essentially black-box processes, would benefit by having effective methods for preserving knowledge for successive generations of training. We develop an approach to preserving knowledge encoded in the hidden layers of several ANN's and collect this knowledge in networks that more effectively make predictions over subdivisions of the entire dataspace. We show that this method has an accuracy advantage over the single-network approach.

We extend the previously developed methodology, adding a non-parametric method for determining transferrable encoded knowledge. We also analyze new datasets, focusing on the ability for models trained in this fashion to be transferred to operating on other storms.

CHAPTER ONE

INTRODUCTION

1.1 Motivation

The surface area of the Earth is 510.1 million square kilometers. The troposphere, which is the layer of the atmosphere where weather takes place, extends from 0 to 12 kilometers above the Earth's surface. This surface area is constantly being scanned by remote sensing devices that are both ground-based and space-based. The scanning technologies are diverse, employing a wide variety of sensors and observing the Earth at varying spatial and temporal resolutions. These instruments produce an enormous volume of data. Therefore, any system making use of such data will *a priori* have to navigate the problem handling of extremely large amounts of data that are only comparable if they are calibrated in exactly the same way or if the differences can be quantified. We may describe the temporal coverage of the Earth as "arrhythmic" since the observations may not happen at a consistent time interval, and "spatially inconsistent" since coverages are spotty and occur using disparate instrumentation.

1.2 Numerical Weather Prediction vs. Empirical Weather Prediction

Numerical Weather Prediction (NWP) models the entire Earth mathematically and predicts future conditions of the atmosphere by extrapolating the current conditions based upon the dictates of the models. It is a precise treatment of the atmosphere and is based on a long legacy of science. There is a growing amount of work that is being done in this direction, which we describe using the term *Empirical Weather Prediction* (EWP). We define EWP as weather prediction by means of analysis of the data alone, disregarding the atmospheric processes that produce this data. In other words EWP deemphasizes knowledge about the physical and chemical processes that describe the atmosphere and relies on machine learning techniques to discover patterns in the data.

Several recent advancements may enable EWP for some critical atmospheric problems. These problems include the determination of the speed and magnitude of wind (wind vector determination) and the extrapolation of atmospheric characteristics into the future (atmospheric condition prediction). These advancements have occurred primarily in the area of remote sensing, which is technically any kind of sensing in which the observer does not have direct contact with that which is being observed. Here we use the term *remote sensing* to mean specifically observations from space using satellite technology. The advancements include new, higher-resolution sensor technology; new satellite constellations; and better ground-based data collections devices. To ensure that the techniques of assimilating, analyzing, and modeling this flood of data keep pace with the development of these collection technologies, high-fidelity, high-resolution simulations are being created and used in anticipation of the deployment of the observation technologies on a grand scale.

As an example, in our research we use an interpolated dataset that has:

- temporal resolution of the data at 15 minutes, meaning that a new snapshot of the entire planet is achieved every 15 minutes;
- 15km grid at Earth's surface (225 km^2). This translates to 2,266,666 grid cells at the surface of the Earth.

- multiple elevations in a vertical profile (normally a vertical profile of 6 cells)

NWP is already computationally expensive at the current resolutions, even over limited areas of the Earth [101]. Because we desire higher temporal and spatial resolutions, it is important to use data-driven EWP to create new prediction models. Since prediction using EWP promises to be less computationally intensive than NWP, it should be able to handle the increased amount of data that results from using these higher resolutions. We do not envision EWP as a replacement to NWP, since NWP is an important endeavor that employs systemic understanding of the atmosphere and atmospheric processes. However, these models will benefit the meteorological community because they will add understanding of the difference in how the atmosphere is understood through the mathematical models and how the atmosphere actually behaves. This will facilitate improvement of their numerical models: a cross pollination between NWP and EWP. It is this finer resolution that now enables the use of data-driven methods to make wind vector determinations as well as predictions of the future state of the atmosphere. The research detailed in this dissertation concentrates on these data-driven methods.

1.3 Contributions

In this dissertation we take advantage of the opportunity exposed by the higher resolution datasets to establish EWP as an alternative to NWP. To do this we will use the attributes of simulated datasets derived from NWP models to identify encodings in deep neural networks for transfer learning in weather prediction. The identification of these encodings (or feature detectors) is an advancement on transfer learning using neural networks because it uses knowledge gained from one area of the dataspace to make assertions about another, spatially distinct area. In developing our technique, we make contributions to the meteorological community as well as to the computer science community.

Our interest in Meteorological data started as a response to the Industry's need for novel ways of using some datasets to enrich others, as in this case where we determine wind vectors from radiometric data. Through the development of our research we discovered that our interest in it is due to its spatial, temporal, functional nature. It is this type of data that we need to develop our techniques for transfer learning. An early problem we address is whether it is possible to determine wind vectors from atmospheric data that does not include previous wind vectors using a lattice of neural networks? This is the domain-specific problem that exposes several computer science research issues due to both the size and the dynamic nature of the problem. We start in this area, with an eye towards increasing the generality of our approach as we develop this research.

In this dissertation we detail our efforts that have culminated in the following contributions:

- Examination of the use of autoencoders to augment radiometric data, giving the system the ability to determine wind vectors. The difficulty is that static "pictures" of the atmosphere cannot be used to determine motion. Wind vector determination is a specific manifestation of this.

- Creation of a method to train neural networks based on stacked autoencoders to perform prediction on spatiotemporal data, whose areas of interest are influenced by the surrounding areas.

- Evaluation of the effectiveness of pre-trained feed forward networks on wind vector determination, and the models' ability to generalize over adjacent spaces. This evaluation will help the community through the application of networks on new areas of the dataspace on which they were not trained, potentially reducing the number of networks needed to make useful predictions over large areas.

- Novel application of unsupervised pre-training across diverse areas of this dataspace in a way that tracks the methods used by NWP.
- A method of achieving a baseline in performance using unsupervised pre-training in an effort to understand how predictions differ for distinct areas of the dataspace.
- Transfer learning analysis of neural networks across the dataspace. We would also like to characterize the transferability of neural network knowledge from one area in the dataspace to another. This analysis concentrates on transferring knowledge between locations that are spatially and temporally different.
- Identification of distinct, transferable encodings in trained neural networks.
- Development of methods for transferring encodings into other networks. This will include a study of the effects of having performed these transfers.
- Development of a highly controlled technique for exploring this dataspace using spatial statistics and functional data registration.
- Development of a method of “divide and transfer” that divides the dataspace into contiguous clusters of application and performs an exchange of useful encodings within these clusters.
- A method for expanding the hidden layers of a network to adapt the architecture to increasing data variety.

- A locality-sensitive clustering procedure that pays attention to both the data in a spatiotemporal dataset as well as feature identifiers that have been collected during a pre-training process.
- Application of this technique across three separate storm events, with transfer between two of them.
- A "cluster to application" mapping procedure for transferring clustered neural network models.

1.4 Organization

This dissertation is organized as follows: Chapter 2 contains background information on weather prediction and neural networks. It also introduces the dataspace and the general set of problems we are intending to address. We describe the NWP-enhanced dataset that we use as ground truth data, the data organization, and the current state of NWP. This chapter also contains a review of literature related to this work. Some basis for the assumptions and the general approach that we take to addressing the problems in this domain are addressed in this chapter. This includes information about spatial and temporal distribution of our data as well as our general decisions regarding neural network architecture.

Chapter 3 presents some results of the first set of experiments we conducted in this area. We show that unsupervised pre-training may be used in a deep-learning setting to estimate wind vector components and that they can approximate the functional relationships between radiometric data and wind vector data.

Our second set of experiments evaluates the spatial generalization of stacked autoencoders and is presented in Chapter 4. In this chapter we begin to explore how well stacked autoencoders can be adapted to new areas of the dataspace whose data were not part of the initial training process. In other words, we see how well a trained model can make predictions about areas that are adjacent to the initial area over which the model was trained.

In Chapter 5 we use spatial statistics to identify transferrable encodings within trained networks. We treat the meteorological data as functional, leveraging functional data registration for a view into how these functions differ.

We continue developing our method of identifying transferrable encodings of the data in Chapter 6. In this chapter we show how we divide the dataspace into clusters that warrant the use of unique neural networks. These clusters exchange internal encodings among each other in a novel subset transfer process. We compare this technique, using several thresholds for the number of clusters produced, with the performance of a single network over the same space.

In Chapter 7 we demonstrate extending the process that was presented in Chapter 6 using a non-parametric process for identifying encodings. We also demonstrate cross-dataset transfer learning, using data from Hurricane Harvey and Hurricane Irma from 2017.

Finally, in Chapter 8 we give a brief summary of our major contributions, list some conclusions, and give directions that this research can take in the future.

CHAPTER TWO

BACKGROUND

2.1 Numerical Weather Prediction

Our area of application, meteorology, is a very developed field. In this section we give an introduction to the state of the art in weather prediction. Numerical Weather Prediction (NWP) involves using large supercomputers to solve a set of non-linear partial differential equations, called the *Primitive Equations*, that govern atmospheric physics [29]. The horizontal motion (momentum) equations for a spherical Earth (Equations 2.1 and 2.2) represent Newton's second law of motion, stating "...the rate of change of momentum of a body is proportional to the resultant force acting on the body, and is in the same direction as the force." [101]

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - \omega\frac{\partial u}{\partial p} + fv\frac{\partial \phi}{\partial x} + F_x \tag{2.1}$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - \omega\frac{\partial v}{\partial p} + fv\frac{\partial \phi}{\partial y} + F_y \tag{2.2}$$

where t is the time, u, v, and ω are the independent Cartesian velocity components, f is the Coriolis parameter, ϕ is the geopotential height, F_x and F_y are friction in the x and y directions, ϕ is the geopotential (gravity-adjusted) height, and p is the barometric pressure. The thermodynamic temperature equation (Equation 2.3) accounts for adiabatic (occurring without loss of heat) and diabatic (occurring with loss of heat) effects on temperature.

$$\frac{\partial T}{\partial t} = -u\frac{\partial v}{\partial t} - v\frac{\partial v}{\partial t} + \omega\left(\frac{RT}{C_p p} - \frac{\partial T}{\partial P}\right) + \frac{H}{C_p} \tag{2.3}$$

where R is the gas constant for air, T is the temperature, C is the thermal capacity, p is the pressure, H is heat and P is the rate of water input through precipitation. The continuity equation (Equation 2.5) for total mass states that mass is neither created nor destroyed. Equation 2.4 is an application of this principle for water vapor.

$$\frac{\partial q}{\partial t} = -u\frac{\partial q}{\partial x} - v\frac{\partial q}{\partial y} - \omega\frac{\partial q}{\partial p} + E - P \tag{2.4}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial p} = 0 \tag{2.5}$$

where q is the specific humidity and E is evaporation. Finally, equation 2.6 represents the ideal gas law.

$$\frac{\partial \phi}{\partial p} = -\frac{RT}{p} \tag{2.6}$$

In addition, further parameterizations must be estimated, including heat, evaporation, precipitation, and friction in all directions. Furthermore, a parameterization scheme for sub-grid scale phenomena must be created. The system is also not a closed system, as the Earth absorbs incoming solar radiation and emits some of this radiation back into space [29, 101].

There is a growing amount of work that is being done in the direction of using machine learning models for weather prediction, which we describe using the term *Empirical Weather Prediction* (EWP). We define EWP as weather prediction by

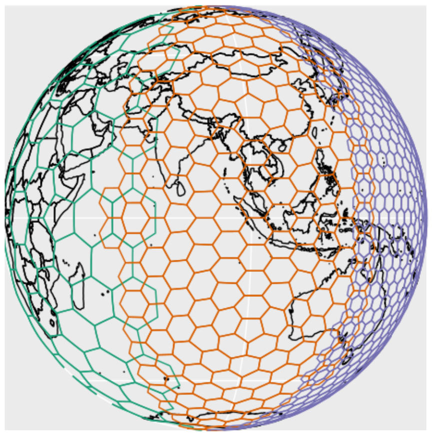


Figure 2.1: Depiction of the Hexagonal Discrete Global Gridding System [15]

means of analysis of the data alone, disregarding the atmospheric processes that produce this data. In other words, EWP deemphasizes knowledge about the physical and chemical processes that describe the atmosphere and relies exclusively on machine learning techniques to discover patterns in the data. To be clear, we do not make any claims that Empirical Weather Prediction (EWP) provides functionality that is superior to NWP. We hypothesize that EWP can supplement the NWP endeavor in many ways and that both methodologies can benefit greatly by paying attention to one another. At the outset, barring any ability to explain why the neural networks that EWP will use make the decisions that they make, EWP can be used to inform NWP practitioners about the effectiveness of their models. Such mechanisms, while beyond the scope of this dissertation, would be very useful avenues for exploration.

2.2 Discrete Global Gridding Systems

NWP uses the Discrete Global Gridding Systems (DGGS) as a method of geographical binning. A DGGS [69] is a system of adjacent polygons that cover the entire planet. Such systems allow the collection of point observations that occur in the same vicinity so that each point collected in a polygon is seen as representative of that polygon. The size of these polygons determines the resolution of the weather information system. Throughout this dissertation we use grid cells with 15 kilometer diameters, so in the language of remote sensing our system can be described as having a resolution of 15 kilometers at sea level.

We have chosen to use a geodesic DGGS grid based on hexagonal grid shapes [36]. All of this formed the basis for our geospatial data binning procedure. Figure 2.1 shows a hexagonal DGGS superimposed over the entire planet at three different resolutions. The geographic location of the point observations coupled with the polygon arrangement makes up the "gridding" procedure for our data.

2.3 Neural Networks

Artificial Neural Networks (ANN) [61] are computational models originally inspired by how the human brain was thought to function [41]. They act in a highly complex, nonlinear, and parallel fashion, as the human brain does, and in distinction to how traditional models of computation behave. To do this they employ a massive interconnection of simple artificial neurons as their constituent computational components. They store and make use of experiential knowledge from data using synaptic weights on the connections between these artificial neurons [41].

Figure 2.2 represents a model of an individual neuron. The inputs on the left of the diagram correspond to neuronal synapses, receiving input from a procedure that feeds data to the network from an incoming dataset. After input is received, the values of the inputs are then multiplied by weights w_m. The circular node in the center that is labeled with the large Σ is a summing junction, whose function is to calculate the dot product of $\mathbf{x}$ and $\mathbf{w}$ plus the bias b. The result is the *activation potential* or *activation* v_k, which is then sent to an activation function, whose job it is

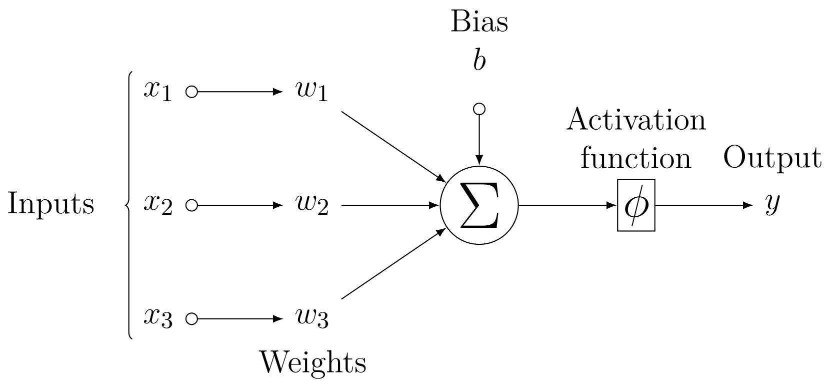


Figure 2.2: Nonlinear model of an artificial neuron [41]

to limit the permissible amplitude range of the output signal to some finite value [41], usually between 0 and 1. We use the ReLU function [39] as our activation function, which forces the activation to be between 0 and 1.

The backpropagation procedure is then used to adjust the weights of each of the connections between the nodes in the network. It does this by comparing the output value of the network with the actual value in the training data. When this difference is calculated, the negative gradient of this value with respect to the weights of the incoming connections is computed. This lets the system know what it needs to do to the incoming weights in order to minimize the prediction error. In multilayer neural networks this procedure is propagated back from the output all the way to the inputs, hence the name "backpropagation."

The operation of this neuron is summarized here:

$$y_k = \phi\left(\sum_{j=1,k}^{m} w_{j,k} x_{j,k} + b_k\right)$$

where m is the number of inputs to the neuron; k is an index identifier for a neuron in the collection of neurons (if multiple neurons are employed); $w_{j,k}$ is the weight of the kth neuron given the jth input; $x_{j,k}$ represents the jth input; b_k represents a "bias"

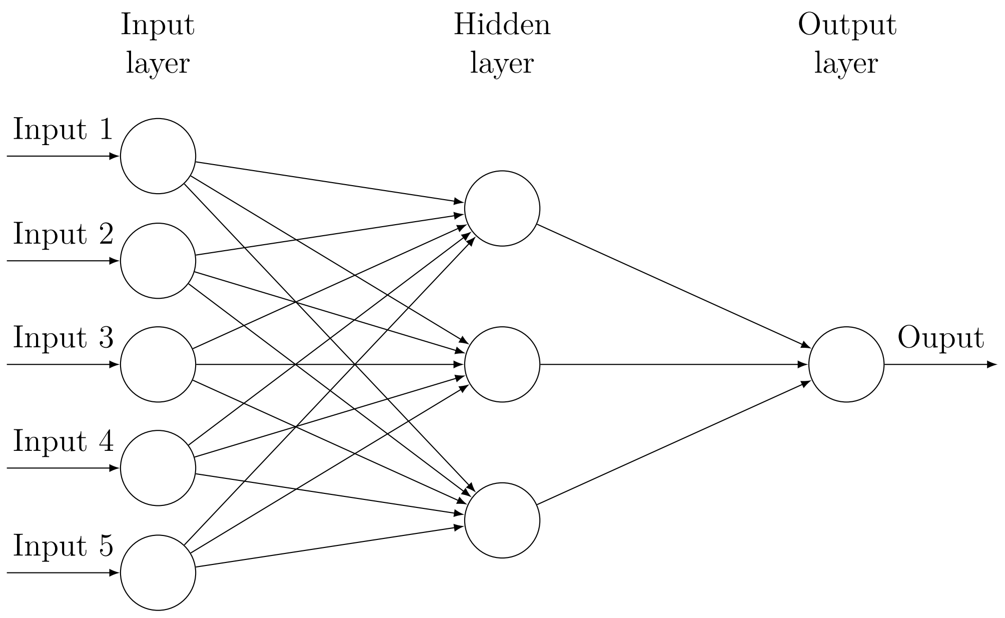


Figure 2.3: Architectural graph of an multilayer feedforward network

node; and $\phi(\cdot)$ is the activation function. This bias node is sometimes included to apply an affine transformation to the value before the activation function is applied.

Common activation functions are the logistic function and the hyperbolic tangent function. For our purposes we used the Rectified Linear Unit (ReLU) function [48], corresponding to the following:

$$ReLU(u) = \begin{cases} 0 \text{ if } u < 0 \\ \\ u \text{ otherwise} \end{cases}$$

Figure 2.3 shows the structure of a multilayer feedforward network with one hidden layer. The middle layer is *hidden* because it is not observed or interacted with directly from the perspective of either the input or output of the network [41]. Hidden layers have the effect of extracting higher order factors that are inherent in the structure of the input data. The outputs from the input layer are used as inputs to the hidden layer. The outputs from the hidden layer are in turn used as inputs to the output layer. In many cases, including in the early parts of this investigation, more than one hidden layer is used.

We use neural networks that are trained using the *backpropagation* algorithm, which is a type of supervised training [81]. It uses the gradient of the error function iteratively to change the weights of each of the inputs to the summing junction. For the error function, we use the Mean Squared Error (MSE), with respect to the weights, as follows:

$$E = \frac{1}{n}\sum_{i=1}^{n}(predicted\_value_i - actual\_value_i)^2$$

where E is the MSE and n is the number of elements that have been predicted.

2.4 Autoencoders and Unsupervised Pre-Training

Autoencoders, whose were first defined by Ackley, Hinton, and Sejnowski [9] are neural networks that encode a vector of inputs into a lower-dimensional representation, with the aim of decoding them into the original dimensionality with minimal distortion [14]. Stacked Autoencoders have been credited with beginning the current enthusiasm for Deep Learning in general [39]. They also form the basis of the neural networks in our experiments.

Figure 2.4 depicts an autoencoder. The features enter into the input layer on the left. The values of these dimensions are then multiplied by their weights and sent to the hidden layer. The activations from the hidden layer are then multiplied by their respective weights and sent to the output layer, which through training will become an approximation of the input layer. Through training, the hidden layer becomes a lower dimensional representation of the input layer, capable of being reconstructed at the output layer. Mathematically, $\mathbf{x}^* = g(f(\mathbf{x}))$ such that $L(\mathbf{x}^*, \mathbf{x})$ is minimized,

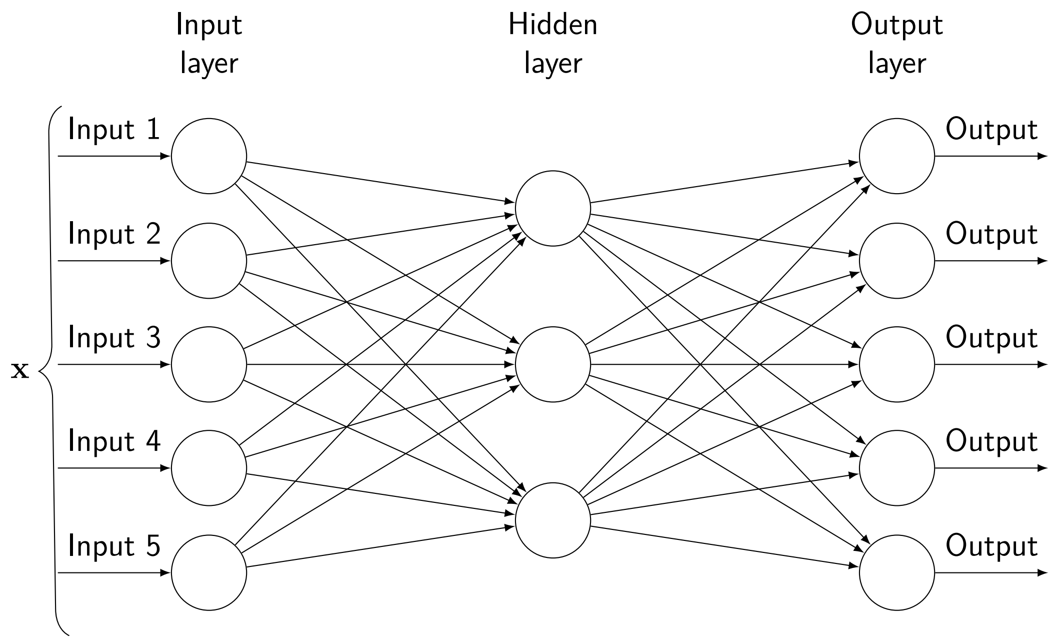


Figure 2.4: Architectural graph of an autoencoder

where L is the loss function, which in the case of autoencoders is the reconstruction error, which itself is defined as:

$$L(\mathbf{x}^*, \mathbf{x}) = ||\mathbf{x} - \mathbf{x}^*||$$

In Unsupervised Pre-Training (UPT), autoencoders are stacked using an iterative procedure in which an autoencoder is trained to capture the behavior of the input data [26]. The supervised learning procedure endeavors to benefit from leveraging an unsupervised learning component during the pre-training phase. In the experimental analysis performed in [26], results are offered that indicate that “unsupervised pre-training guides the learning towards basins of attraction of minima that support better generalization from the training data set.”

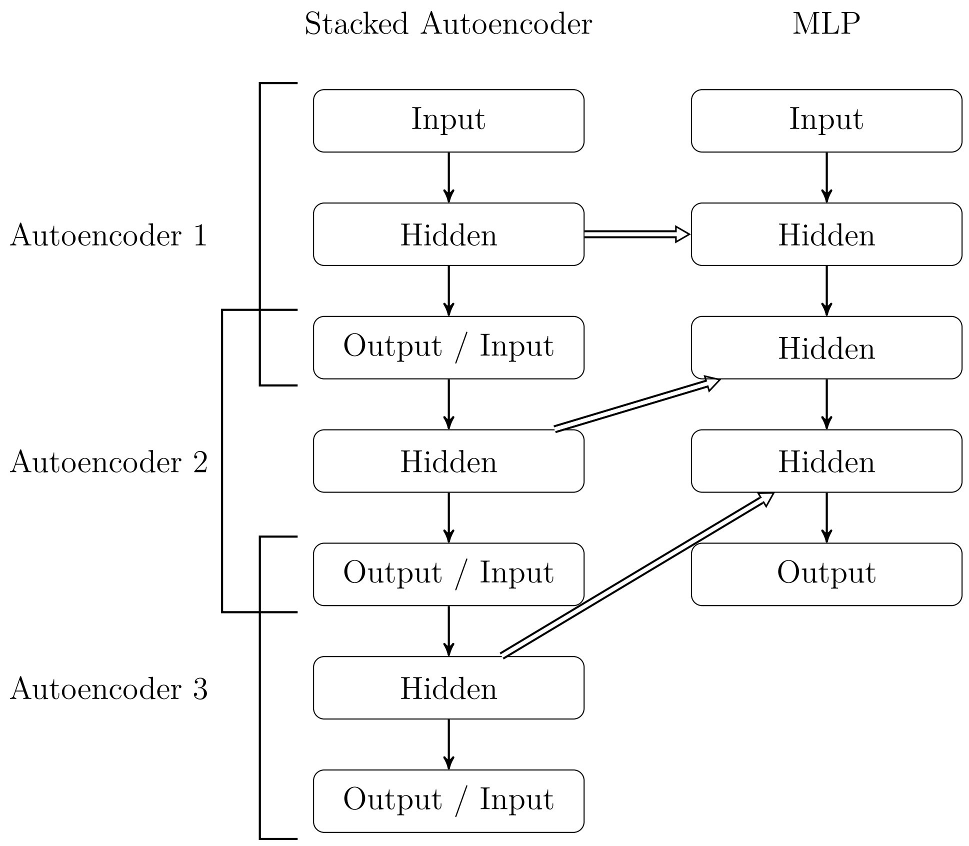


Figure 2.5: General unsupervised pre-training operation

Figure 2.5 shows the operation of UPT when stacking three autoencoders. The segment labeled “Autoencoder 1” at the top-left of the figure shows the input-hidden-output layers of a single autoencoder like the one that is depicted in Figure 2.4. After training Autoencoder 1, the hidden layer is extracted and added to the set of hidden layers in a Multilayer Perceptron (MLP), as the arrow from the “Hidden” layer in Autoencoder 1 to the “Hidden” layer under the MLP column shows. The same procedure is repeated for Autoencoder 2, using the encoded output values from the hidden layer of Autoencoder 1 as input, and Autoencoder 3 using the encoded output values from the hidden layer of Autoencoder 2 as input. The final layer of the stacked autoencoder should be a representation that is as close to the Autoencoder 2 output representation as possible and is therefore labeled “Output / Input.” To complete the construction of the MLP, the same input layer from Autoencoder 1 is added to the MLP at the head of the network. This is so we can train the MLP on the same training dataset and with the same dimensionality that we used to train Autoencoder 1. The output layer of the MLP consists of the desired number of nodes that correspond to the classifications, or a single node in the case of regression.

2.5 Neural Networks in Weather Prediction

After recognizing that weather data is nonlinear and follows irregular trends, Abhishek, et al. [8] developed “effective and reliable” nonlinear models using ANNs to simulate them. In [8] they recognized “massive computational power required to solve the equations that describe the atmosphere, error involved in measuring the initial conditions, and an incomplete understanding of atmospheric processes” as factors motivating the ANN approach. Further, they observed that “the ANN minimizes the error using various algorithms and gives us a predicted value which is nearly equal to the actual value.” Their approach specifically addressed temperature forecasting and used an assemblage of temperature data, year over year, to predict a recent daily maximum temperature. A key way that we differ from this approach is that we do not rely on the data from the same time of year in the previous years for our predictions. We rely on temporally contiguous data instances during the lifetime of a hurricane.

2.6 Weather Data

2.6.1 Space-Borne Remote Sensing

There are several problems that come from the areas of meteorology and remote sensing that expose a need for EWP and for this research. This section introduces these problems and explains why these issues persist today.

"Remote sensing" refers to any observation that an instrument makes without making direct contact with the subject of the observation [106]. Here, we define "remote sensing" strictly to be the use of space-bourne observation technologies.

2.6.2 Passive Microwave Radiometry

The primary type of remote sensing data we use for our experiments is Passive Microwave Radiometry, which operates on the principle that the Earth is a black-body emitter of radiation. Some of this radiation is in the form of thermal emissions, which at longer wavelengths can be detected using instruments called Passive Microwave Radiometers (PMR) [106]. The detection of phenomena at these wavelengths is important because some of them interact with the constituent components of the atmosphere, such as nitrogen, oxygen, and water vapor. For example, radio waves at 118 Ghz interact with oxygen [87].

Figure 2.6 depicts the interaction of microwaves with oxygen and water vapor between 0 GHz and 500 GHz. The areas where the transmission percent dips to 0 percent are called *absorption bands.* Microwaves with a wavelength of around 118 GHz interact with water, as can be seen in the chart where the transmission percent dips to zero and then returns to around 74 percent. Likewise, microwaves at a wavelength of around 183 GHz interact with water vapor. It is this interaction that allows radiometers to measure the state of these chemicals in the atmosphere.

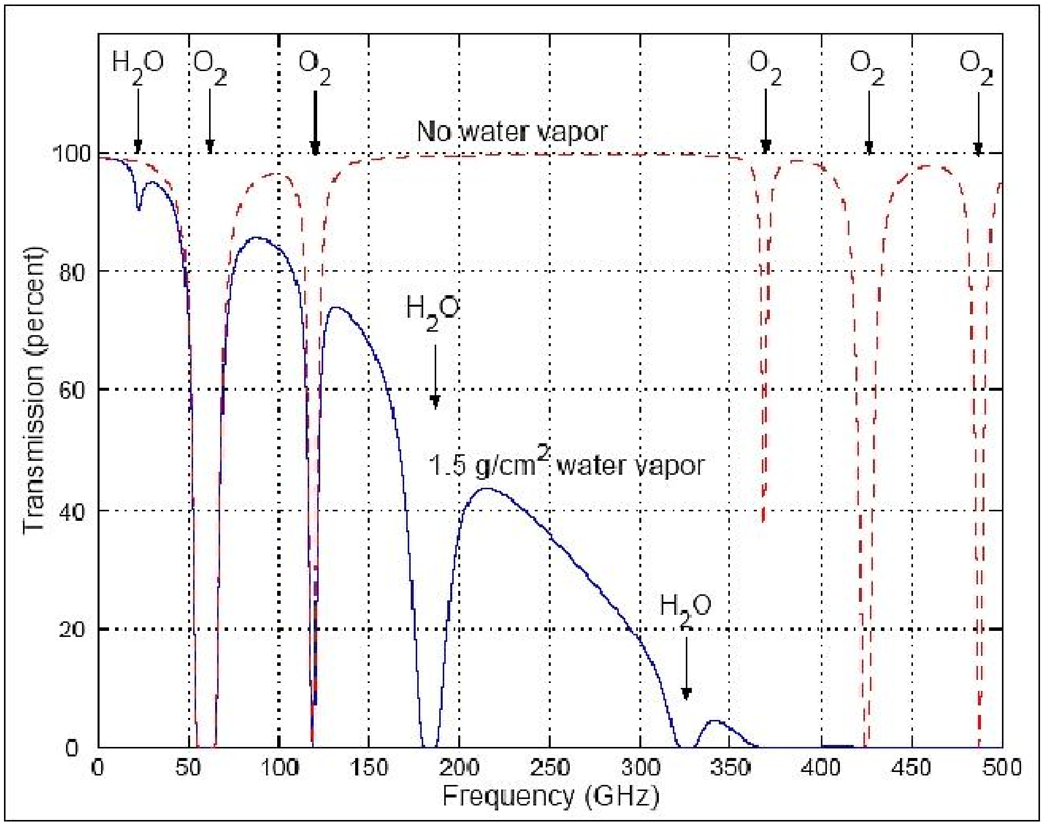


Figure 2.6: Chemical absorption of microwaves [4]

Figure 2.7 is a composite image taken by the Advanced Technology Microwave Sounder (ATMS) on the Suomi National Polar-orbiting Partnership (NPP) satellite [97]. The image shows the state of the atmosphere as seen through the instrument at 183.31 GHz. Due to various constraints on the system (power, orbital profile, etc.) such an image can only be formed a few times a day. Since the temporal resolution is so low, the types of effective machine learning that can be done using such data is very limited because weather phenomena can form and dissipate on a time scale that is much finer than the resolution of this system. An example of this phenomenon is summer storms in the midwest that can form and dissipate in the span of merely a couple of hours. Observations taken before and after a summer storm cause valuable

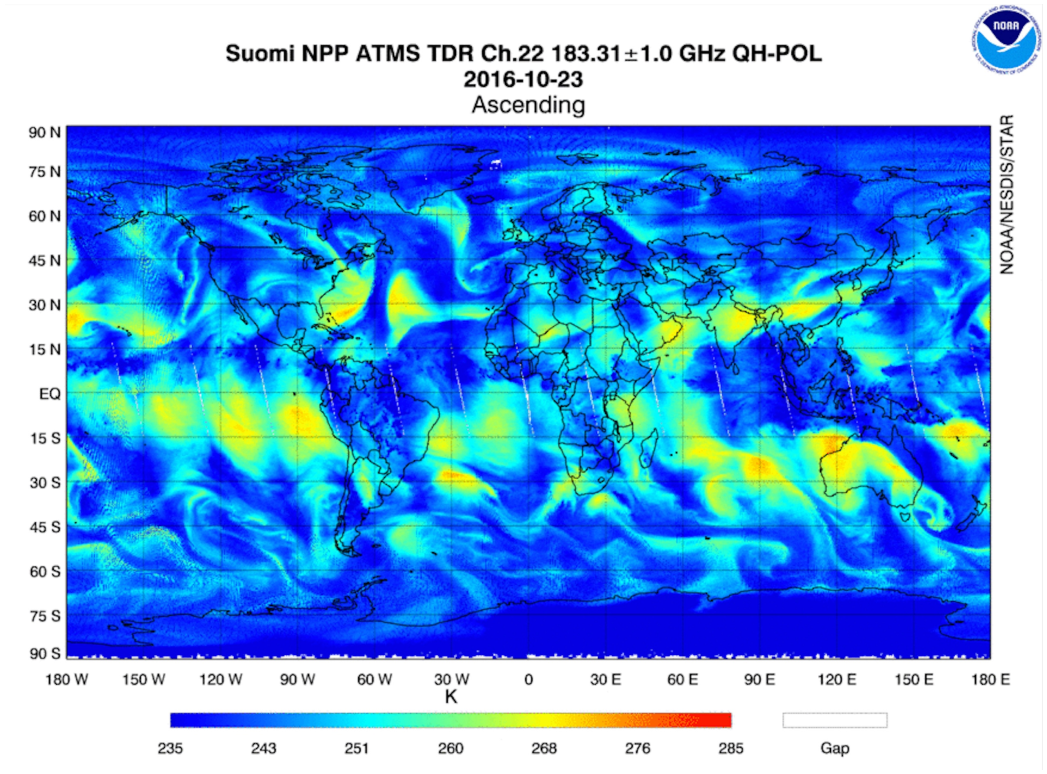


Figure 2.7: Global view at 183.31 GHz [97]

information that would be missed completely as the storm is not represented in the data at all.

On either side of 118 GHz there are channels that are relatively opaque or transparent[1]. In other words, if the sensitivity of the radiometer is modulated to a wavelength slightly lower than 118 Ghz then the radiometer will be able to see through more of the atmosphere and closer to the surface of the Earth. If it is modulated slightly higher, then it is not able to see as far into the atmosphere and will therefore be reading information about higher altitudes. The effect of this is that if several channels surrounding 118 GHz are read, then a vertical profile of the atmosphere can be obtained [34]. Such a vertical profile is called a "sounding" because

[1]The exact level of opacity depends on other factors.

of an analogy that is made between obtaining “soundings” of water depth using sonar. The number of frequencies that are used will provide a “vertical resolution,” which is the spatial resolution of the readings in the vertical direction. In effect, this produces a 3-dimensional representation of the Earth’s atmosphere.

2.6.3 Wind Data

In the existing data and literature, wind vectors are represented using three components: the East-West component u, the North-South component v, and the vertical component w. A negative u value indicates eastward motion and positive indicates westward motion, a negative v value indicates southward motion and positive indicates northward, and a negative w value indicates downward motion and positive indicates upward.

Information about wind vectors primarily comes from ground stations that are located throughout locations on the surface of the Earth. This data is quite limited as it can only give information about the two dimensional wind vector at exactly one altitude. This information is supplemented by wind vector readings that can be obtained using Doppler radar and weather balloons, which can take wind measurements aloft, but only in specific areas.

Space-borne wind vector data can be derived using scatterometer measurements [64]. This is an active technology, meaning that the instrument broadcasts a signal to the subject of observation and senses the reflection of that signal. Because it is active, it requires a lot of power to run and therefore has seen limited deployment.

2.7 Weather Determination and Prediction

We define *weather determination* to mean the specification of weather dimensions in the current time from other dimensions that are co-located in space and time. In other words, if we are determining wind vectors, we determine them from dimensions that do not include wind vectors. We define *weather prediction* to mean extrapolation of weather dimensions into the future. This section introduces the problems upon which we focus in this area.

2.7.1 Wind From Radiometric Readings

Establishing a wind vector measurement using only radiometric readings is difficult and is currently a frontier in weather analysis [33]. Radiometers do not detect movement. Nevertheless, knowledge of wind vector measurements using only radiometric readings could be very important, since it might be able to add vertical wind vector soundings to a large portion of the Earth's atmosphere [99] and increase the awareness of the disposition of wind over the rest of the Earth at a minimal cost.

2.7.2 Problem with Neural Networks Meteorology

Mathematical precision can be computationally expensive. Since this mathematical precision is desired, very large supercomputers have to be dedicated to the problem of weather prediction [29]. This is because of the complexity of the equations that are used to describe the atmosphere and the computational techniques that have to be utilized to perform the extrapolations necessary to predict the weather (see Section 2.1) [8]. Still, imprecision creeps back into the system at every level, since there are many causes of uncertainty in a system as complex as the weather.

In some cases where remote sensing is used for weather observation, the types of observations that can be made are limited by the instrumentation. This is the case when all that is available is satellite observations, such as when observing oceans, or other remote areas of the Earth. Using the example of radiometric data, it is easy for this type of instrument to determine things like temperature, moisture content, and precipitation type, which are all critical to determining atmospheric conditions. But other useful information, such as wind vectors, must be inferred because it cannot be observed directly using this technology.

2.8 Literature Review

2.8.1 Stacked Autoencoders

One purpose of using Autoencoders is to obtain a lower-dimensional encoding of a dataset [9]. Stacked autoencoders are arranged so that a progressively lower dimensional representation of each successive layer is achieved by autoencoding the previous layer. The general concept is that each layer captures the fundamentally important features of the previous layer. As in generalized ANNs, in stacking these Autoencoders, there is an assumption that the data expresses a hierarchical nature. Exploiting this, each subsequent layer learns a higher level of abstraction from its predecessor. Pre-training an ANN in this way results in a set of ANN layers that have captured the main variations in the input data [26].

2.8.2 Numerical Weather Prediction

Numerical Weather Prediction (NWP) began in its basic form about a century ago [56]. Previously, weather forecasting relied on the opinions and experience of experts using very sparse observations with data points drawn crudely, by hand, on weather maps. It was at this time that the principal concern of "advection," the transport of fluid characteristics and properties by the movement of the fluid itself, was identified. It was the job of the human forecaster to perform the necessary weather prediction extrapolations using this sparse and irregular data [29].

A significant advancement came about with the development of thermodynamics and enumeration of physical principles that describe the atmosphere. A purely mathematical approach was proposed by Abbe in [6]. Explicit scientific analysis was undertaken following this by Bjerknes [19], who enumerated basic variables of pressure, temperature, density, humidity, and the three wind vector components as basic variables to be used in prediction. He also identified equations of motion, continuity, and state expressing thermodynamics for these variables. Bjerknes' efforts were hampered by the absence of computational capability. Modern approaches to weather prediction are current incarnations of the methods of Richardson [101], who favored methods of mathematical extrapolation following the dynamics identified by Bjerknes.

Drawing on his interest in turbulent fluid flows, John Von Neumann recognized that the weather prediction problem was ideal for an automatic computer [56]. The effort that came out of this interest identified the problem that integrating the primitive equations using a short time step rendered computation intractable.

Numerical methods in meteorology were created because computers, while excellent at arithmetic, had difficulties with calculus, especially in this domain [57]. The equations of Bjerkens, and of those who advanced the science in subsequent years, require computers to approximate solutions to a variety of differential equations. Often the time scales on which the approximations to the solutions to these differential equations occur dictate the precision with which phenomena can be predicted [96].

A current state of the art regional weather forecasting model is the Weather Research and Forecasting model (WRF). It is a portable, open-source community model that allows the inclusion of custom physics packages [110]. We made use of WRF in our research to perform critical data interpolations that gave us consistent temporal and spatial resolutions.

2.8.3 Non-neural Machine Learning Methods in Weather Prediction

2.8.3.1 Overview Machine learning is becoming increasingly prevalent in weather prediction. A manifestation of this is IBM's acquisition of the product and technology assets of The Weather Company in 2016 and subsequent launch of "Deep Thunder," which is "the World's Most Advanced Hyper-Local Weather Forecasting Model" [2]. In several press releases, IBM has said that they are using historical weather datasets to train machine learning models, though IBM and The Weather Company are not revealing the exact machine learning methods that they are using [20, 45, 46].

In general, a variety of statistical methods have been used to analyze weather data and to create predictive models. Machine learning, in particular, is becoming more popular as an avenue of research. Williams et al. provide an example using decision trees by combining NWP forecasting information with random forests in order to rank the performance of predictors [52]. Their intent was to find the best combination of predictors based on day, hour, and location, because the optimal set of predictors may change due to daily initial conditions. Using the data from several variables obtained from satellite and ground station measurements, combined with the ground truth provided by TITAN (an external process for thunderstorm boundary detection) [25], they successfully identified the best predictors for thunderstorm initiation, which simplified the extrapolations that had to be performed in the NWP process.

2.8.3.2 Genetic Programming Feng, et al. created ClimateLearn [28], which is a toolkit for "climate network analysis." "Climate networks" are networks of correlated fields on a global or regional level. The correlations are characterized by similar patterns of climate variability. They used genetic programming to forecast the time evolution of NINO3.4 index (the sea surface temperature anomaly averaged over the area of interest) which is a scalar characteristic of El Niño. The scalar characteristic would give them information about the presence and intensity of El Niño events.

An interesting technique was created by Singh and Gill, who concentrated on time series prediction, assuming a temporal relationship in the state of the dimensions and also a relationship among the dimensions [90]. This was an adaptation of the work of Siu, et al., who improved the backpropagation algorithm using genetic programming as a strategy of escaping local minima [92]. Such relationships were not considered in the work of Siu, et al., who were not using weather data in their experimentation.

2.8.3.3 Probabilistic Graphical Models Cofiño, et al. use probabilistic graphical models to perform meteorological time series prediction by modeling spatial and temporal dependencies among weather stations [22]. They derived a Bayesian network whose graph was learned from the data available from these weather stations. They proposed an efficient methodology for obtaining precipitation forecasts using inference mechanisms on the Bayesian networks created using the temporal dependencies extracted from the weather station dependency models.

2.8.3.4 Support Vector Machines Using support vector machines, Rao, et al. [77] estimated the maximum temperature of a day based on the maximum temperature from a series of previous days. They used linear support vector regression to make this prediction. Their approach did not differ significantly from the approach of Radhika and Shashi [73], who achieved better performance, as measured by the mean squared error, than a multilayer perceptron by minimizing an upper bound on the generalization error rather than the training error. Pérez-Vega, et al. [70] made an additional attempt at the temperature forecasting problem using support vector machines, though they emphasized the preprocessing of the data as the critical step in performing the regression.

In 2011, Wei used support vector machines to build a system for forecasting hourly precipitation during typhoon events [102]. For this, Wei used the advanced wavelet kernel because of the kernel's high predictive ability when applied to areas of pattern recognition. The performance of this technique was compared with a more traditional support vector machine that used a Gaussian radial basis function. They found that the use of the the wavelet kernel resulted in slightly better prediction accuracy than when the Gaussian radial basis function was used.

2.8.4 Deep Learning Methods in Weather Prediction

Subhajini provides a survey of the application of neural networks in weather forecasting [95]. The conclusion of the survey is that reasonable accuracy can be obtained using a variety of neural network models. The audience of this work is, in general, meteorologists and Earth scientists. This is an example of several survey papers that have been published in the last few years, signaling a positive change in attitudes towards neural methods in this domain.

Deep methods have seen a trend in usage in meteorology, particularly for rainfall prediction. Hernández, et al. [42] used deep learning to predict accumulated daily precipitation a day in advance. They used a "multilayer autoencoder" (as opposed to a stacked autoencoder), where the hidden layers of the autoencoders are directly connected to single nodes in the single hidden layer of a multilayer perceptron. They used a variety of input variables related to rainfall such as relative humidity, temperature, dew point, rainfall, sun brightness, and barometric pressure over ranges varying from the previous three to the previous five days. Their technique outperformed the use of multilayer perceptrons alone and significantly outperformed the work of Abhishek et al. [7] who used a multilayer perceptron three layers deep with 10 and 20 nodes per layer. Our primary methodology differs from that of Hernández, et al. because we are using unsupervised pre-training to train stacked autoencoders rather than to condition the nodes of a single layer perceptron.

Salman, Kanigoro, and Heryadi compared the prediction performances of recurrent neural networks, conditional restricted Boltzmann machines, and convolutional networks in the exploration of hierarchical weather representations that result from the training of these networks [83]. Specifically, they wanted to predict the behavior of El Niño / Southern Oscillation (ENSO) parameters (wind, oscillation index, sea surface temperature, and outgoing long-wave radiation). Their study was preliminary, so their methodology was largely exploratory, and they did not claim that they had a superior technique.

Grover, Kapoor, and Horvitz studied combining discriminatively trained predictive models with a deep neural network that models the joint statistics of weather related variables [40]. They made use of the spatial characteristics of the data, learning long-range spatial dependencies in the process. They used this technique effectively to predict wind speeds, dew points, geopotential heights, and temperature over a period of time ranging from 6 to 24 hours. Their technique performed well against some state of the art methods that are currently being used at the National Oceanic and Atmospheric Administration (NOAA).

In 2015, Dalto, Matsuko, and Vasak [24] applied deep ANNs to the problem of ultra-short-term wind forecasting. By intelligently reducing the number of input variables, they allowed the network to complete training in a reasonable amount of time. This input variable reduction was based on Partial Mutual Information Input Variable Selection, which determines the degree of redundancy of input variables.

They used separate pre-trained Stacked Denoising Autoencoders for each wind vector component, much like components of our work.

Multilayer pre-trained Restricted Boltzmann Machines were used in the work of Narejo and Pasero [63] to perform “meteonowcasting,” (i.e., predicting conditions of the present or very near future). They used another method based on mutual information for parameter optimization relative to the time-step of interest. Using this analysis, the critical atmospheric parameters for making these very short term predictions were found to be the temperature and the time of day. The authors used several different network topologies, adapting the architectures to the diverse configurations of atmospheric parameter variables.

Wang, Balaprakash, and Kotamarthi used output from the Weather Research and Forecast model (WRF) to train deep neural networks in hopes of finding “...an accurate alternative to physics-based parameterizations.” [100] They concentrated specifically on the Planetary Boundary Layer, which is the “portion of the atmosphere in which the flow field is strongly influenced directly by interaction with the surface of the Earth.” [68] In this area, the wind velocity, temperature, and humidity dynamics greatly influence the conditions of the underlying atmosphere. They achieved a successful simulation of vertical profiles, or how the aforementioned variables behave moving vertically from the surface, within the boundary layer for several response variables, including wind velocity, temperature, and water vapor, successfully accounting for the diurnal cycle. This is another verification of the viability of ANN’s in this domain. Our work is quite different in that we are searching for transferable encodings within these trained networks.

2.8.5 Transfer Learning

There are many times in machine learning when it is assumed that data upon which machine learning procedures can be trained, tested, and applied are drawn from the same dataset, implying that they have the same distribution. This is not always the case. Sometimes the assumption that there are commonalities in the way source and target datasets behave. By "sufficient" we mean that the training will apply to the domain of interest to a useful degree. Transfer Learning (TL) makes use of knowledge and internal representations that were captured from auxilliary domains to build models for new domains [55, 67, 98, 104]. In TL, a machine learning system uses some information that has been learned in one setting to improve performance in another setting [39].

According to Pan and Yang [67], there are three major sub-disciplines that make up TL: inductive transfer learning, unsupervised transfer learning, and transductive transfer learning. In inductive transfer learning, the source and target data domains are the same, but the source and target tasks are different. This is used in situations where source and target labels are available, or in a situation where the source domain labels are unavailable but the target domain labels are available. Transfer in the latter situation is constrained to representation learning, which helps in the target domain since the domains are the same. Unsupervised transfer learning, which is an important subtask of our research, is similar to inductive transfer learning but differs in that what is being sought is a clustering, density estimation, or dimensionality reduction. It is used when the source and target domains and tasks are all different but related. In our research we are primarily interested in transductive transfer learning, in which the source and target domains are different but related, as in different segments of the atmosphere (due to differences in underlying terrain, storm lifecycle, solar radiation, etc.), and the source and target tasks are the same.

An advance in TL was made by Cao, et al. [21] who were looking at the problem of how to avoid "negative transfer," where the transfer of a trained procedure to a new domain hinders prediction performance. Cao, et al. used a Gaussian Process to adapt TL schemes, automatically estimating the similarity of the source and target tasks in a process they called "safe transfer." Such an approach is interesting from our perspective as we hope to adapt the predictor to the problem, and not necessarily the problem space to the predictor.

More recently, Hu, Zhang, and Zhou [44] addressed the short-term wind speed prediction problem for situations in which wind speed data over new wind farms was unavailable. The authors were able to effectively predict wind speed by considering the terrain and topography of the areas of interest along with the prerequisite atmospheric variables.

2.8.6 Explainable AI

Addressing a deep concern that is an obstacle for the adoption of ANNs in many domains, Samek, Wiegand, and Muller [85] state that "it is not clear what information in the input data makes them arrive at their decisions." Methods that combine the worlds of transfer learning and explainable AI have been elusive. In the community, published work that addresses this issue is sparse at best.

Through discovering useful and transferable encodings in the internal representations within ANN's we believe that we are addressing the explainable AI question pertaining to ANN's on their own terms. In order to create an explanation of how an ANN makes the decisions that it makes we would have to translate the endemic logic of ANN's to logic that humans can understand. This translation will be imprecise. If we can get ANN's to exchange information amongst themselves according to their own logic we will have made a significant contribution to this area of understanding.

2.8.7 Functional Data Analysis

Functional Data Analysis (FDA) is the process of analyzing data that is generated as a result of some underlying process. Prerequisite to this treatment, the data are modeled as a function, often of time or space. One of the first texts to define FDA and describe several attending analysis methods is written by Silverman, et al. [89]. FDA treats functions as single entities, rather than sequences of individual observations. To do this, the data are represented as linear combinations of basis functions, which are well suited for storing information about functions. Silverman, et al. consider two major types of basis functions: Fourier basis for describing periodic data and B-spline for data without strongly cyclic variation. The power of matrix algebra could be brought to bear on the resulting linear combinations. In our experiments we use the B-spline basis, since in our FDA treatment of data we are studying intra-day measurements. In the future, for multi-day datasets cyclic variation may become more of an issue because of the diurnal cycle. This may then motivate the use of a Fourier basis for FDA.

Bel, Bar-Hen, and Petit used FDA to "...account for interactions between climate variables..." in their study of paleoecological data [16]. They were studying the impact of global warming on genetic diversity and how climate affects this. Their functional dataset reconstructed climate variables (temperature and precipitation) over the past 15,000 years. Their work is an example of how FDA techniques have helped Earth scientists to discover patterns and interactions among variables over a long term.

In meteorological Earth science, King [50] used functional analytic methods to analyze climate change data. The author fit temperature time series data to a spline basis to track temperature changes in US cities over the last few decades. This, however, was not a machine learning study.

2.9 Technical Approach

In this research we use neural networks whose hidden layers are autoencoders that have been preconditioned using unsupervised pre-training. We adapt the internal layers to the data and then proceed with training the model as a feedforward network. This chapter gives some reasons for our treatment of the data and our use of autoencoders around which to build our models.

2.9.1 General Data Treatment

The primary data in which we are interested for the purposes of this dissertation is spatial, temporal, functional data. Meteorological data fits this description because it is highly structured data whose dynamics are dependent on time and space relationships. We call this data “functional” because we treat the process that generated the data as a singular entity, rather than treating the data as sequences of individual observations. The functions in meteorology have space and time as their abscissas. Importantly, these temporal and spatial dependencies are consistent across time and space. This means that in an ideal collection of this data there is a regular, consistent temporal interval of data distributed at collection points that are consistent distances from one another.

Meteorology is a subject that is experiencing a lot of excitement regarding artificial neural networks and associated technologies. Circumstances are not always ideal though, as the distribution of data collection points is not evenly spaced, nor is data collected at a consistent time interval. This motivates the use of the Weather Research and Forecasting model, which is a system that makes spatial and temporal interpolations based on numerical models.

2.9.2 Use of Autoencoders

Right now there is a wide variety of ANN types that are being leveraged in meteorological research. For example, Wei opted for a “Deep Learning-based DNN” in his study on wind simulations, with a reference to the use of Convolutional Neural Networks (CNN’s) to expand the capabilities of their model [103]. Wimmers and Velden instead used “DeepMicroNet,” which is based on CNN’s as well [107]. The spatial and temporal nature of this data might motivate the use of CNN’s, which by design address both space and time, and Long Short Term Memory (LSTM) networks, which by design address time. In this dissertation we elected to use feedforward networks composed of autoencoders that were pre-trained using unsupervised pre-training because, for now, we wanted to create as direct a relationship with the state of the art of (NWP) as possible.

To explain our model’s relationship to NWP, Figure 2.8a shows the *forecast domain*, which consists of 3-dimensional grid cells. The figure shows rectangular cells, which represent just one among a multitude of ways the atmosphere can be divided into grid cells. Figure 2.8b shows an example of one dynamical process in which adjacent grid cells have an effect on the conditions of the grid cell about which predictions are being made. The physics underlying the grid cell and its surrounding grid cells are illustrated in Figure 2.8c, where some more detail is given about the types of processes that are exchanged among the cell under study and its surroundings.

Our version of this situation employs similar logic, considering a center cell and each of this cell’s surrounding cells. Figure 2.9 shows the hexagonal gridding system that we used. Each of the cells surrounding the center cell in this figure is analogous to the conditions in the cells surrounding the forecast domain cell in Figure 2.8. We elected to simulate the general spatial treatment of NWP in our ANN environment in the manner that we did because the relationship to the NWP spatial treatment is very

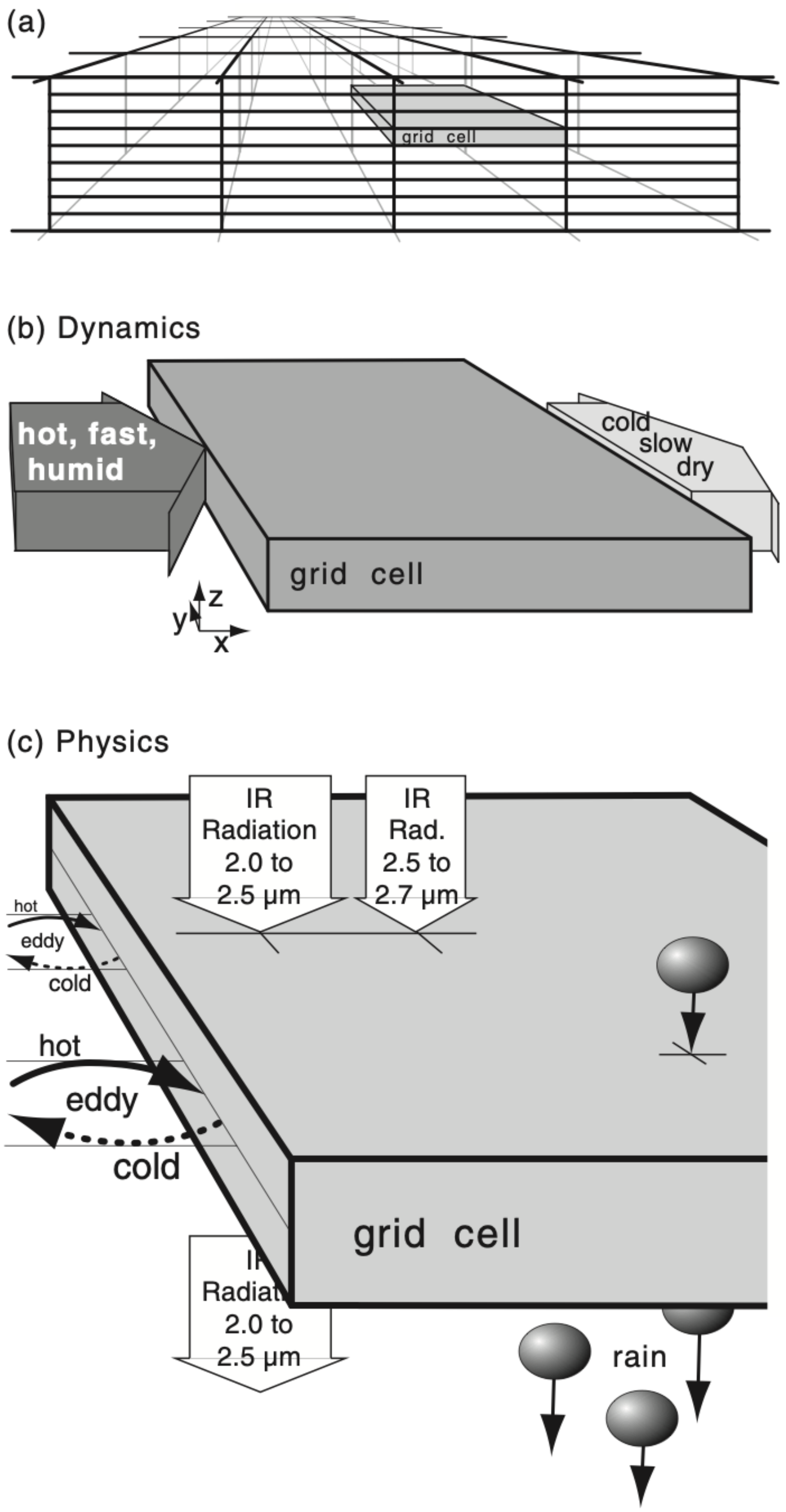


Figure 2.8: Forecast domain representation [94]

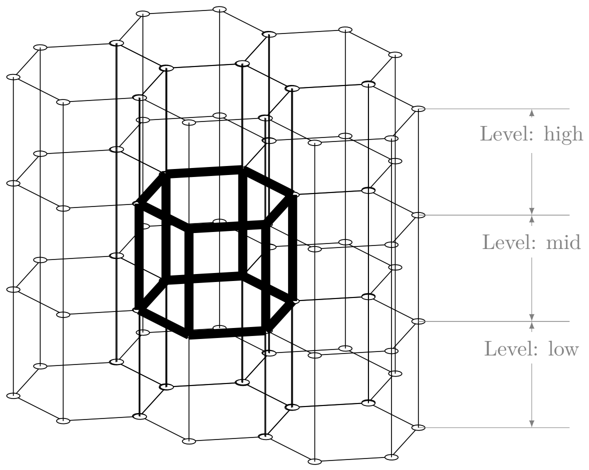


Figure 2.9: The 27 3D DGGS Cell Lattice

straightforward. Since this relationship has been established here, the adaptation to Neural Network models more adapted to spatial and temporal data can proceed in the next generation of this research.

2.10 General Data Pipeline

Our general data pipeline is shown in Figure 2.10. Significant variations on this general procedure will be explained in the chapters that follow as they happen. The raw data was taken from satellites, ground stations, and other collections that normally act as inputs for the WRF system. Radiometric readings and wind readings came from separate systems, as different instruments are used to collect either one of them. The design matrices were formed as multidimensional arrays. Normalization was done on a per-dimension basis across all time slices and all locations to a range of between -1 and 1. The negative values only happened in the case of the wind vectors (it does not make sense to have negative precipitation, for example). The

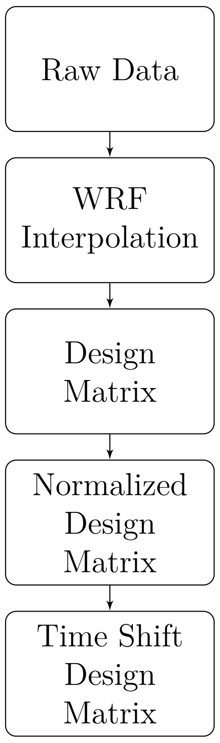


Figure 2.10: General data pipeline

time shifts were performed so that we could account for the temporal nature of the system within our chosen architecture, that of feedforward networks.

2.11 General Experimental Design

Figure 2.11 depicts our general experimental design. Again, clarifications are made in each of the following chapters where the experimental design differs from this. For most of our experiments we use a single prototype network from which all of our networks are derived. This is so the initial network will have the same initialization values, removing a source of random variation. This is not the case for the initial experiments. The design matrix is the output from the general procedure that is explained in Section 2.10. The training procedures are subject to variation

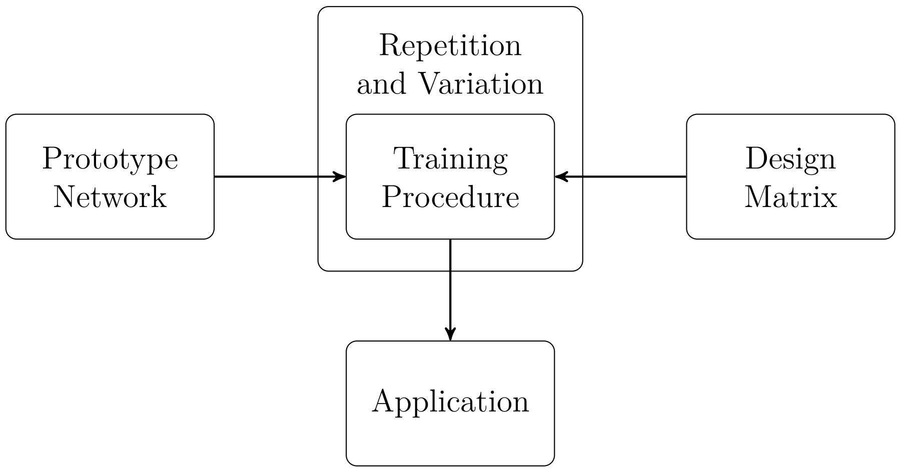


Figure 2.11: General experimental design

and one or more instances of repetition, which will be explained in the chapters that follow. Finally, the application (testing) of these models happens after these other procedures have been completed.

CHAPTER THREE

DEEP LEARNING FOR WIND VECTOR PREDICTION

3.1 Overview

Figure 3.1 shows Hurricane Sandy as it was on October 29, 2012. This hurricane was the second costliest storm in the history of the United States [27]. We used data collected during this hurricane, coupled with WRF interpolations of the conditions during this hurricane as our medium for our first investigation in the first phase of the research detailed in this dissertation.

For this phase we used unsupervised pre-training to condition a four-layer multi layer perceptron to determine wind vectors using radiometric data. The reasons that we used the four-layer architecture are that we believed that this architecture provides enough of an abstraction from the 168 input dimensions (for no-wind experiments) and still retains enough differentiation to allow us to examine how the training process proceeded.

3.2 Dataset

The output of the WRF model interpolations of Hurricane Sandy were created by Zhang and Gasiewski [109]. It consists of data captured directly from the hurricane plus temporal and spatial interpolations. The WRF-enhanced dataset provides data at a 5 km grid spacing and generates the entire field at a time interval of 15 minutes. Figure 3.2 shows the area covered by the interpolated dataset, which corresponds to an area of the Eastern seaboard of the United States for one 24 hour period. The area is bound in the southwest at 26.4902° north, 81.6064° east and in the northeast

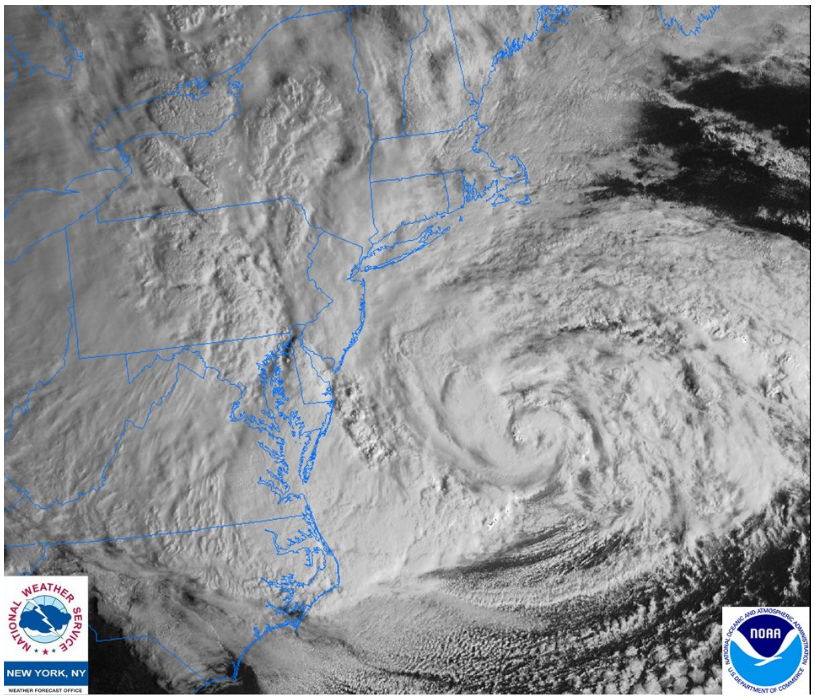


Figure 3.1: Hurricane Sandy [27]

at 41.2117° north, 60.3809° east. The box in Figure 3.2 appears distorted due to the geodesics on the surface of a sphere.

Figure 3.3a shows where the standard deviation of the barometric pressure varied most, with the darker areas corresponding to a greater standard deviation and the lighter areas corresponding to a lesser standard deviation. This area was in the northeast of the dataspace. Likewise, in Figure 3.3b the lighter areas represent areas where the standard deviation of the temperature over the entire 24 hour dataset was lower and the darker areas show where it was higher.

After we chose a point in the Earth's atmosphere to analyze, we collected information about all of the areas surrounding this point. The dataset was binned using a 3D hexagonal DGGS, so we made predictions concerning the entire cell in which our point of interest lies. Each individual cell has a horizontal resolution

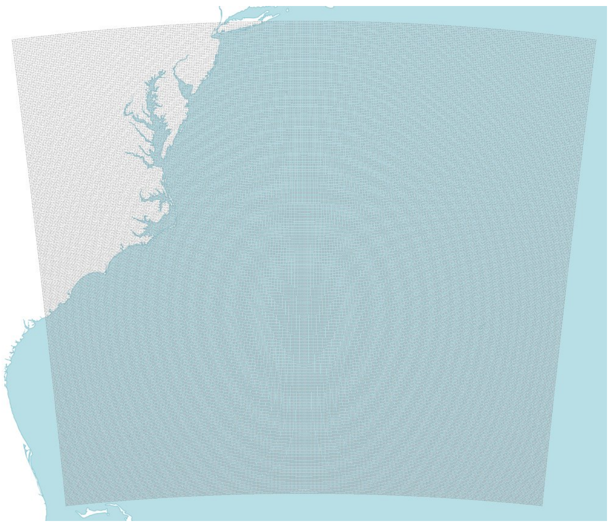


Figure 3.2: The area of interest included in the WRF simulations

of 15 kilometers at the surface of the Earth. Figure 3.4a shows this lattice from the surface of the Earth to the top of the atmosphere. Each level of this lattice represents around 2,000 meters. We are interested in the sixth level from the ground, since this corresponds to a level of around 30,000 to 36,000 feet, which is an area of heavy air traffic.

Figure 3.4b shows a cutaway of this lattice. The cell at the center contains the point in the atmosphere in which we are interested. There are three layers of surrounding cells from which we collect data in order to perform our experiments. These correspond to the elevation containing our point of interest and the cells that lie immediately above and below this point. For each of these elevations, data from the center cell and all of the cells corresponding to all six hexagonal azimuths are collected. The number of data points for each cell, and for each time period was on the order of 10 (varying by at most 2 in either direction).

In this phase of our research, our major concern was to ensure that feedforward networks with unsupervised-pre-trained could be used to simulate some of functions

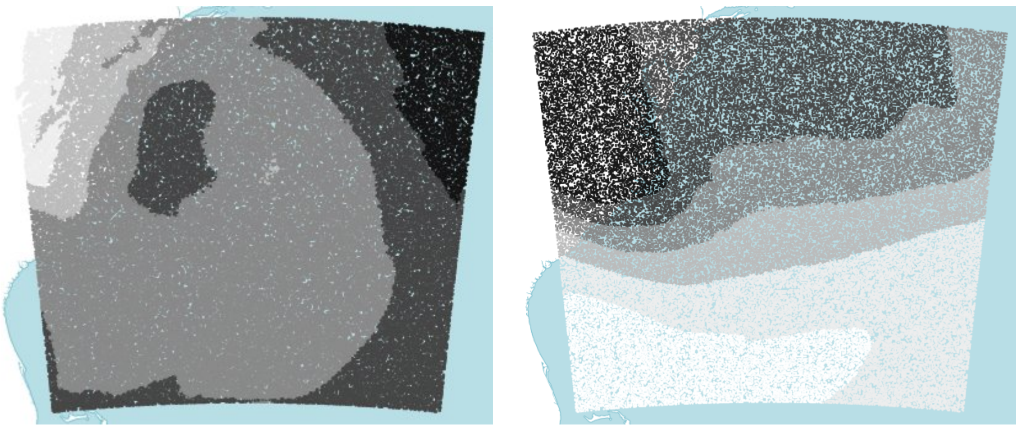

(a) Barometric Pressure (b) Temperature

Figure 3.3: Standard deviations over time of barometric pressure and temperature

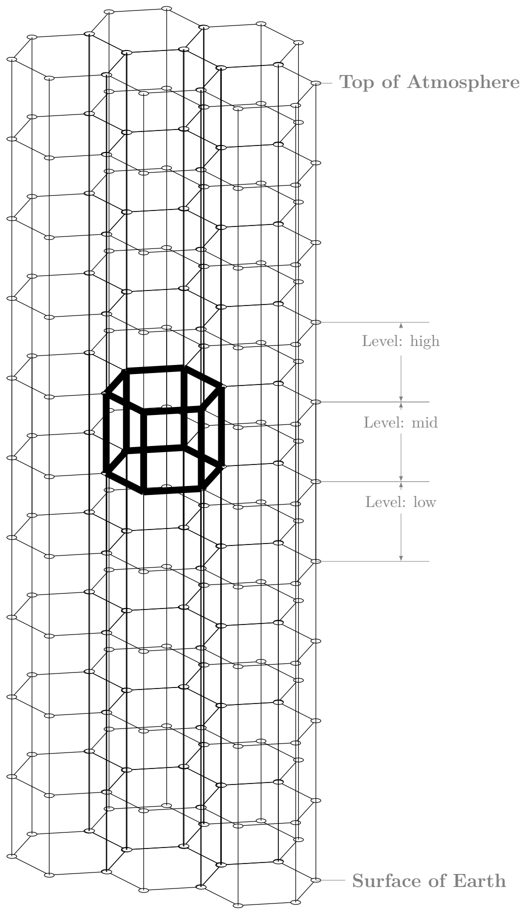
that spatially relate atmospheric components, specifically relating to radiometric data and wind vectors. This is useful in meteorology because of the elusive nature of the specific functional relationship between radiometric data and wind vector determination, as radiometers (in their existing deployments) do not detect wind. We believed that these relationships could be learned from the data that is available using this neural model.

Our experiments all have the assumption that the current weather conditions of any location depends on:

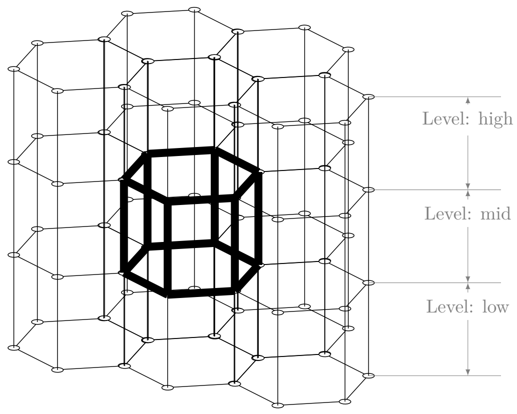
1. the previous state of that location;
2. the previous states of the locations immediately surrounding that location.

This is in keeping with the explanation of our experiments' relationship to NWP explained in Section 2.9.2. With respect to our experiments, if we are attempting to determine wind vectors for a location, the current conditions can be influenced by:

1. the current state of that location not including the wind vector (which we do not know);



a) Hexagonal lattice from ground to top of atmosphere



b) The 21 3D DGGS cell lattice

Figure 3.4: DGGS and surrounding cells

2. the previous state of that location;

3. the previous states of the locations immediately surrounding that location.

3.3 Approach

Referring again to Figure 3.4, we are interested in making predictions about the center cell, which is outlined in bold. As stated above, we assume that the current conditions in that cell are influenced directly by its own conditions in the previous

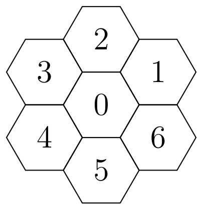


Figure 3.5: 7 cells representing one DGGS level

time slice, and the conditions of all the cells adjacent to it in the previous time slice. The figure puts this concept into three dimensions, showing that influences on the cell in which we are interested can come from any azimuth and above and below.

To simplify, Figure 3.5 depicts one level of the DGGS lattice. In this case we want to infer the condition of cell 0. Cells 1 through 6 correspond to adjacent cells on this level with their respective azimuths. For example, cells 1 through 6 in one scenario would have azimuths 32, 78, 139, 215, 260, and 319, indicating that all of these cells surround the center cell.

For each experiment, we retrieved all of the data for each variable and for each time slice from each cell for the entire duration of the simulation. These readings were sorted by time. Each reading, including the wind vector reading, was regarded as representative of the entire cell from which it came. Since our DGGS cell resolution is 15 km and the simulation resolution was 5 km, there were on the order of 10 readings (sometimes varying by as much as 2) for each time slice for each cell. Therefore, in most cases, the radiometric data do not correspond to the exact locations of the wind vector data, but they always occur in the same cell.

3.4 Experiments

Figure 3.6 shows the general outline and process flow of our experimental procedure. This section explains each of the steps of this procedure in detail.

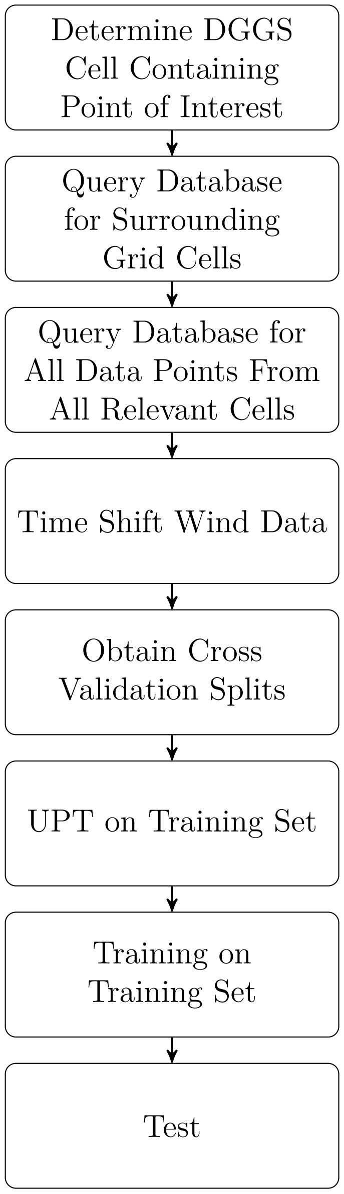

Determine DGGS Cell Containing Point of Interest
Query Database for Surrounding Grid Cells
Query Database for All Data Points From All Relevant Cells
Time Shift Wind Data
Obtain Cross Validation Splits
UPT on Training Set
Training on Training Set
Test

Figure 3.6: Overview of the experimental procedure

3.4.1 Locate Containing DGGS Cell and Retrieve Surrounding DGGS Cells

For each of our points of interest on the map, a PostGIS query is made to the database to obtain the DGGS cell that contains this point and another set of queries to obtain all of the cells surrounding this cell. For each of these cells, their azimuths

Table 3.1: Data dimensions for each point in the simulation

Reading Source	Reading Name
Radiometry Simulation	Temperature
	Pressure
	Cloud Density
	Rain Density
	Ice Density
	Snow Density
	Graupel Density
Wind Simulation	Wind U
	Wind V
	Wind W

are recorded and will be used later to identify the location of individual readings. The height information is not held in the DGGS data structures, though in a future incarnation of this system this could certainly be the case.

3.4.2 Retrieve Cell Data Points

For each of the 22 cells retrieved from the database[1], all of the points containing data are retrieved. Table 3.1 shows the dimensions of data at each of these points. The radiometry data provides the temperature, barometric pressure, cloud density, and precipitation type. The WRF wind interpolations provide Wind U, Wind V, and Wind W which are the u, v, and w wind vector components. These are the points of data that we are interested in for our prediction, where u is the east / west vector component, v the north / south component, and w is the vertical component. Each of these readings are tagged with the elevation and azimuth information from the cell

[1]Included in the cells retrieved is the data from the 21 cells for the previous time slice plus another query for the current conditions of the center cell, making 22 cells. If the altitude of interest is at sea level or at the highest altitude captured by the dataset, the bottom-most or top-most level does not exist. So the data from the 14 cells in the previous time slice plus the current conditions of the center cell makes 15 cells.

Table 3.2: Retrieved design matrix

Time Slice	Input Dimension 1	Input Dimension 2	$\cdots$	Input Dimension n	Output Dimension 1	Output Dimension 2	Output Dimension 3
1	$x_{1,1}$	$x_{1,2}$	$\cdots$	$x_{1,n}$	$y_{1,1}$	$y_{1,2}$	$y_{1,3}$
2	$x_{2,1}$	$x_{2,2}$	$\cdots$	$x_{2,n}$	$y_{2,1}$	$y_{2,2}$	$y_{2,3}$
3	$x_{3,1}$	$x_{3,2}$	$\cdots$	$x_{3,n}$	$y_{3,1}$	$y_{3,2}$	$y_{3,3}$
4	$x_{4,1}$	$x_{4,2}$	$\cdots$	$x_{4,n}$	$y_{4,1}$	$y_{4,2}$	$y_{4,3}$
5	$x_{5,1}$	$x_{5,2}$	$\cdots$	$x_{5,n}$	$y_{5,1}$	$y_{5,2}$	$y_{5,3}$

Table 3.3: Time shifted design matrix

Time Slice	Input Dimension 1	Input Dimension 2	$\cdots$	Input Dimension n	Output Dimension 1	Output Dimension 2	Output Dimension 3
1	$x_{1,1}$	$x_{1,2}$	$\cdots$	$x_{1,n}$	$y_{2,1}$	$y_{2,2}$	$y_{2,3}$
2	$x_{2,1}$	$x_{2,2}$	$\cdots$	$x_{2,n}$	$y_{3,1}$	$y_{3,2}$	$y_{3,3}$
3	$x_{3,1}$	$x_{3,2}$	$\cdots$	$x_{3,n}$	$y_{4,1}$	$y_{4,2}$	$y_{4,3}$
4	$x_{4,1}$	$x_{4,2}$	$\cdots$	$x_{4,n}$	$y_{5,1}$	$y_{5,2}$	$y_{5,3}$
5	$x_{5,1}$	$x_{5,2}$	$\cdots$	$x_{5,n}$			

in which the reading was taken. For example, the temperature for the high-level cell at azimuth 260 becomes the dimension azimuth_260_high_temperature.

3.4.3 Time Shift Wind Data

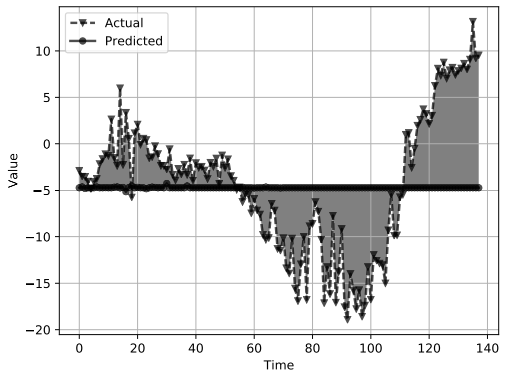
After the previous step we have a set of data that is arranged as in Table 3.2. Input Dimensions 1 through n represent the radiometric dimensions we are using to predict the wind vectors for all of the cells shown in Figure 3.4. Output Dimensions 1, 2, and 3 represent the wind vector components that we are trying to predict. We use one single-output network for each output dimension.

For each of the input dimensions, and as was mentioned in Section 3.3 we are interested using the conditions of the surrounding cells in the previous time step to determine the conditions of the center cell at the current time step. To represent this to our procedure we shift the output dimensions by one time step, creating a new design matrix with the output dimensions of the current time step associated with the input dimensions of the previous time step. This is shown in Figure 3.3, where the output dimension columns are shifted up by one time slice. A consequence of this is that the output dimension values for the first time step ($y_{1,1}$ and $y_{1,2}$) are no longer used because they have dropped off. Also, the input dimension values for the last time step ($x_{5,1}$ through $y_{5,n}$) are no longer used because they are not associated with output dimensions for that time step any more.

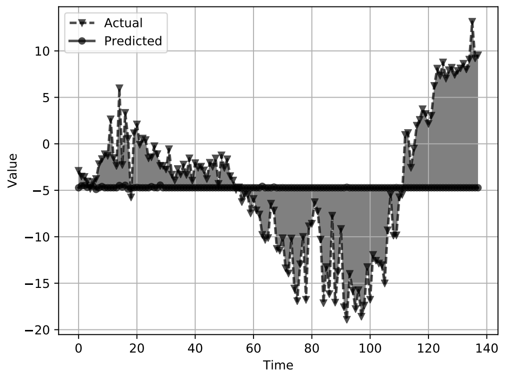
3.4.4 Unsupervised Pre-Training on Training Set

For unsupervised pre-training we used four layers of autoencoders, each with 100 nodes. We determined that this was a reasonable number by running experiments on different sizes of the layers, from 10 through 100, 500, and then 1000. The predicted values began to match the distributions of the holdout data when the networks had around 100 nodes per layer. Figure 3.7 shows the progression of layer sizes from 30 through 60 (10 and 20 were very similar to 30) and Figure 3.8 shows the same for 70 through 100. As can be seen, something happens between 50 and 60 hidden nodes per layer. The reason has yet to be determined and could be a future avenue of research.

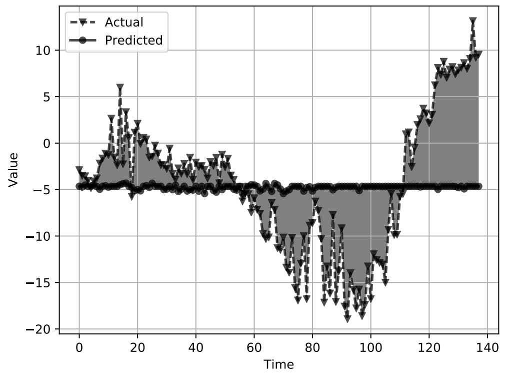
Figure 3.9 shows that using layer sizes of 500 and 1000 hidden nodes per layer produced results similar to those where layers of size 100 were used. From this we concluded that 100 was a reasonable number of neurons in each layer. Typically, in order to work properly, autoencoders need to reduce the inputs to a representation of lower dimensionality, thus creating a bottleneck that summarizes the input data. With layers of size 100 we are constraining the layers to a lower dimensionality, since the input data that includes wind vectors from the previous time slice has 231 dimensions and the input data that does not include wind vectors from the previous time slice has 168 dimensions. If we were to use hidden layers with sizes greater than



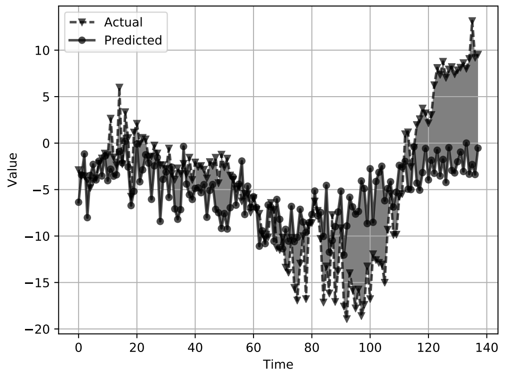
(a) Four Layers of 30



(b) Four Layers of 40



(c) Four Layers of 50



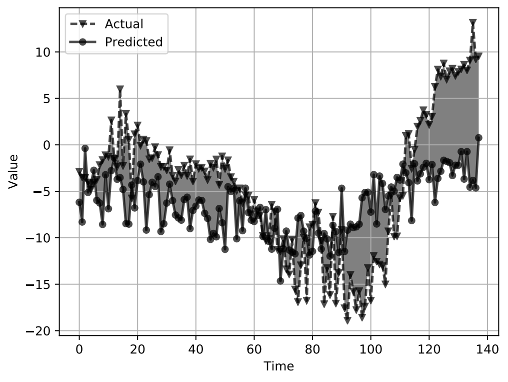
(d) Four Layers of 60

Figure 3.7: Results without wind vectors from the previous time slice using layer sizes of 30 through 60

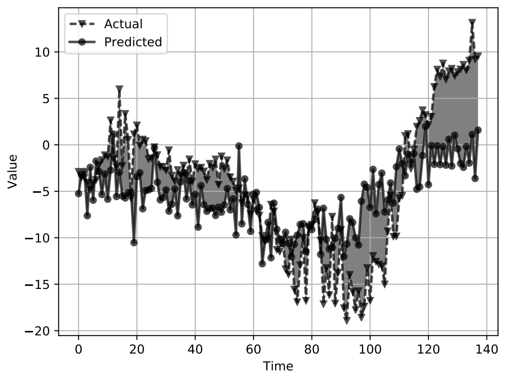
the number of dimensions we would have to employ a regularization procedure to prevent the model from eventually just passing the data through the model without creating any interesting internal representations. While this is interesting in the future we considered this to be beyond the scope of this particular set of experiments.

3.4.5 Neural Network Training

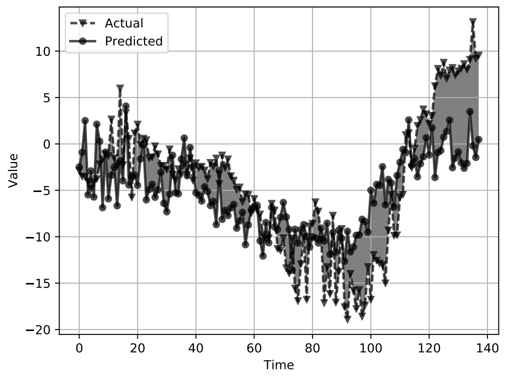
After the pre-training and assembly the network has one input layer accepting all of the data from each dimension, several hidden layers consisting of the hidden



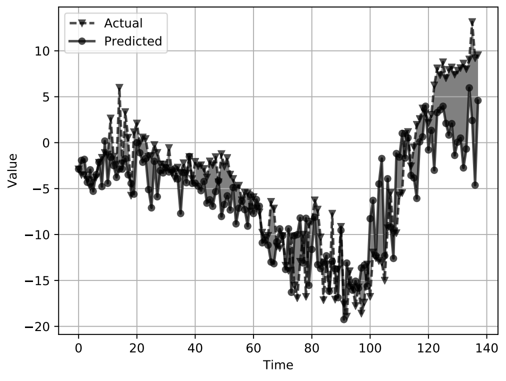
(e) Four Layers of 70



(f) Four Layers of 80



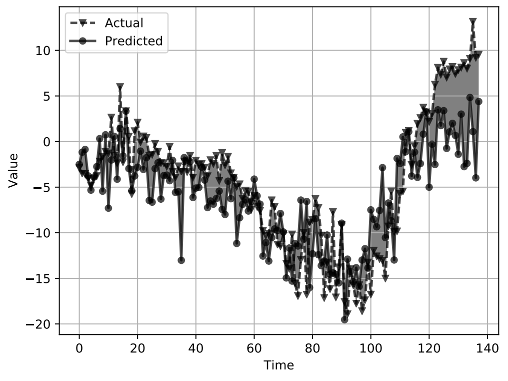
(g) Four Layers of 90



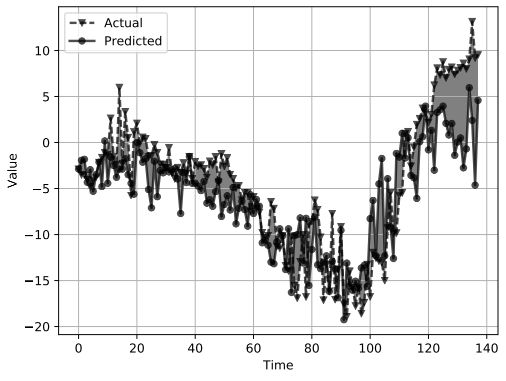
(h) Four Layers of 100

Figure 3.8: Results without wind vectors from the previous time slice using layer sizes of 70 through 100

layers that were trained using UPT, and an output layer consisting of a single node. Since we are using only one output node per network the training and testing for each wind vector proceeded independently. In other words, we constructed one network for the u component, one for the v component, and a third for the w component. Training of this assembled network then proceeded via backpropagation. We did this because, according to [33], these three components should be treated as if they are independent. Preliminary experiments where we combined all components into



(a) Four Layers of 500



(b) Four Layers of 1000

Figure 3.9: Results without wind vectors from the previous time slice using layer sizes of 500 and 1000

a single network supported this recommendation. Future iterations of this research could use one network with three outputs.

3.4.6 Test

We divided the data based on 10-fold cross validation of the time steps. 90% of the data was used for training, which includes training subsets for training (90% of the 90%) and verification (10% of the 90%). Following the fine tuning described in the previous step, we use the 10% test fold to test the performance of our models.

Unfortunately, since we are the only ones using this data in this manner, and it is a new dataset with the unique characteristics that enable this type of analysis and prediction, we are unable to compare our results directly to the results of anyone else. However, we do provide a pioneering investigation into the use of this data using machine learning techniques, and we plan to build upon this knowledge.

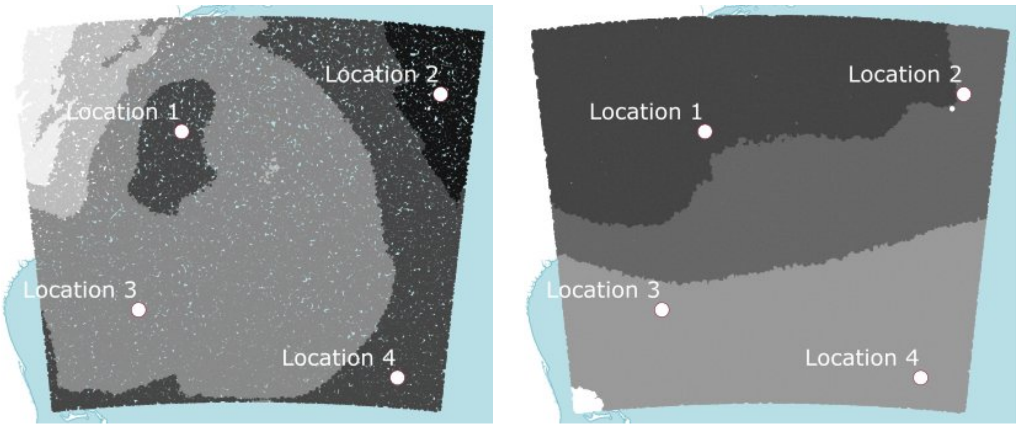


Figure 3.10: Four locations for analysis were chosen at four different areas of varying standard deviation for barometric pressure as well as a variety of standard deviations for temperature. a) Standard Deviation for the Barometric Pressure. b) Standard Deviation for the Temperature. Darker areas indicate higher standard deviation and lighter areas indicate lower standard deviation.

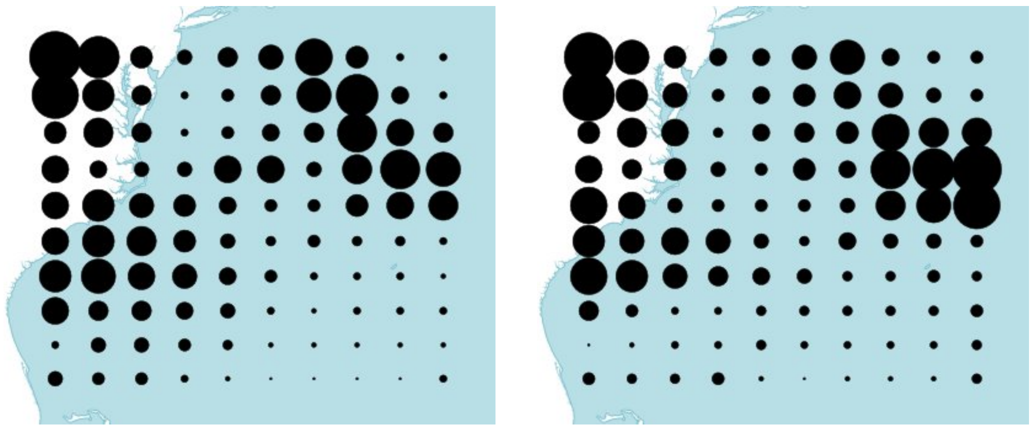
3.5 Experimental Design

3.5.1 Distribution of Locations For Analysis

We chose four points of interest to demonstrate the effectiveness of this methodology. Figure 3.10 shows these points. Our intent in choosing these points was to capture the greatest variety in the underlying weather conditions. Hurricane systems are normally characterized by dramatic swings in barometric pressure. Considering this, the points were chosen, primarily, on the basis of capturing a variety of barometric pressure variation profiles. Of secondary importance, the points also were chosen so that a variety of temperature variation profiles were represented.

3.5.2 Wind Vector Measurements from the Previous Time Slice

For comparison, we wanted to evaluate how having knowledge of all of the previous time slices' wind vector measurements influenced the predictions that



(a) RMSE without Wind (b) RMSE with Wind

Figure 3.11: Heat map for the root mean squared error of all 100 points of interest

were made. In the main thread of experimentation we excluded all wind vector measurements, relying solely on radiometric data for wind vector prediction. For the other thread we included all wind vector measurements for each cell except for the cell about which we were making predictions.

3.6 Results

A heat map of the average root mean squared error (RMSE) across all 10 folds of all of the predictions for each of the 100 experimental locations is shown in Figure 3.11. The smaller dots represent lower RMSE and the larger ones represent higher RMSE. It is interesting that there are areas in which knowledge of the wind vector values seemed to be correlated with higher RMSE than without knowledge of the wind vector. This occurs at the extreme right of the area of observation. But generally, both depictions appear to be similar.

Table 3.4 shows the root mean squared error (RMSE) of predictions for all three vector components for the locations studied, concentrating on the four closely

Table 3.4: Deep network RMSE for the locations studied

	u component		v component		w component	
	With Wind	Without Wind	With Wind	Without Wind	With Wind	Without Wind
Location 1	0.0794	0.1042	0.1218	0.1894	0.1590	0.1022
Location 2	0.1688	0.1505	0.0631	0.1121	0.1415	0.1523
Location 3	0.0115	0.0596	0.1031	0.1284	0.1103	0.1324
Location 4	0.1048	0.1245	0.1736	0.2107	0.1373	0.1241
Avg Over All 100	0.0850	0.1235	0.0958	0.1326	0.1203	0.1359

Table 3.5: Coefficient of determination (R^2) of deep networks for locations studied

	u component		v component		w component	
	With Wind	Without Wind	With Wind	Without Wind	With Wind	Without Wind
Location 1	0.4630	0.3300	0.4558	0.1737	-2.1930	-0.5132
Location 2	0.0769	0.0627	0.7621	0.5473	-0.0673	-0.1679
Location 3	0.9606	0.7963	0.4401	0.3246	0.0818	-0.0407
Location 4	0.5161	0.4617	0.1710	0.0286	-0.1488	-0.1368
Avg Over All 100	0.5956	0.4265	0.5462	0.3888	-0.1931	-0.3395

inspected locations with an average over the k folds. It also shows an average RMSE over all 100 locations studied. Table 3.5 shows the coefficient of determination values (R^2) for these same locations with an average R^2 value over all 100 locations studied. Here

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2}$$

where f_i is the ith predicted value from the neural net, y_i is the ith target (observed) value, and $\bar{y}$ is the mean target (observed) value over the range of prediction.

The results from Location 1 in Figure 3.10 are represented in Figures 3.12, 3.13, and 3.14. Figure 3.10 shows this as being an area of both greater barometric pressure and temperature variability. This may explain why we see wind vector functions that vary more dramatically. It appears in the figure that the predicted values generally simulate the distribution of the actual values. It is apparent from this that the difference between actual and predicted values is smaller when the previous time slice's wind vector information is used for prediction.

Location 2, whose results are shown in Figures 3.15, 3.16, and 3.17 is an area where the standard deviation of the barometric pressure is high and the standard deviation for the temperature is less so. Both the predicted and actual values display less erratic movement from the beginning of the time period through the end. Again, the predictions appear to follow the general shape of the distribution of the actual values and follow the actual values more closely than in the case of Location 1.

Location 3, an area of lower standard deviations for both barometric pressure and temperature, is shown in Figures 3.18, 3.19, and 3.20. It is an area of higher overall wind speed and apparently exhibits a greater degree of predictability. In this area, there appears to be less of a difference between using the wind vector measurements from the previous time slice and not using them.

Finally, the results from Location 4 are represented in Figures 3.21, 3.22, and 3.23. This is an area where the barometric pressure varies more than the temperature. Here we see similar behavior to the other locations, where the predictions made by the networks tend to be smoother than the actual values. It remains to be seen whether the outliers in these cases are artifacts of instrumentation or of added noise in the simulation, if they are not values truly representative of the condition of the atmosphere.

3.7 Discussion and Contributions

We observe that a multilayer perceptron whose hidden layers have been preconditioned using unsupervised pre-training is able to approximate the distribution of wind vector components using nothing but radiometric data. We also observe that the predictions tend to have a smoothing effect on the outliers in the data, though we do not know exactly how to characterize the outlying observations in the first place. An analysis should be performed on the procedure that generated the simulation with the intent of finding out if these outliers are calibration artifacts of the original instrumentation or if noise was added into the simulation to bring it closer to reality or to a desirable test scenario.

Our procedure results in predictions that generally follow the trend of the observations, but are smoothed to a great degree. We do not yet know if this is desirable. There may be many sources of noise that creep into data acquisition or simulation. This, again, is a question that is beyond the scope of this dissertation with respect to the noise attribution. We can only expect an approximation of the functions that describe the forces acting in the atmosphere, without being able to account for aberrations in equipment or simulations. Questions regarding the desirability of the smoothing effect must be understood through further collaboration with the meteorological community.

As a contribution, in this chapter we have developed a method to train neural networks based on stacked autoencoders to perform prediction with spatiotemporal data where the value of the prediction is influenced by the data in spatial areas immediately surrounding the primary area of interest. Approaches to this problem normally involve using convolutional neural networks or long short-term memory networks [39]. However, our method is inspired by the way that the spatial arrangement in the cells and the temporal arrangement of the data instances are performed in numerical weather prediction. The exception is that it is not tied to the physics behind such an approach. As a result, our method can be applied more broadly.

An example of a system with this same assumption is in NWP, where the dependence on previous time steps is limited to one time step. A difference is that our method is limited to considering immediately adjacent cells only, while NWP considers more distant cells with an influence that diminishes with the distance from the cell being considered [94]. Examples of areas for broader application are in neurological data from functional near-infrared spectroscopy (fNIRS) or functional magnetic resonance imagery (fMRI). The set of inputs in this type of data is all numerical or real-valued, is tied to spatial locations, is given as a time series, and can be restricted to the first-order Markov assumption [94]. In addition, the response variable is often real-valued. Any function that can be expressed in this manner can be simulated within a spatio-temporal lattice such as the one used in our approach.

Another contribution is that our method produces an approximated function that produces predictions that follow a smoothed version of the ground truth function. In this respect there is an analogy to be drawn between what our method does and what a denoising autoencoder does. But rather than attempting to remove the noise from the input signal, our approach focuses on reducing the effects of the noise on the prediction. The benefit of this is that we are able to focus on obtaining accurate predictions from the model, instead of denoising the observed measurements.

3.8 Conclusions and Future Work

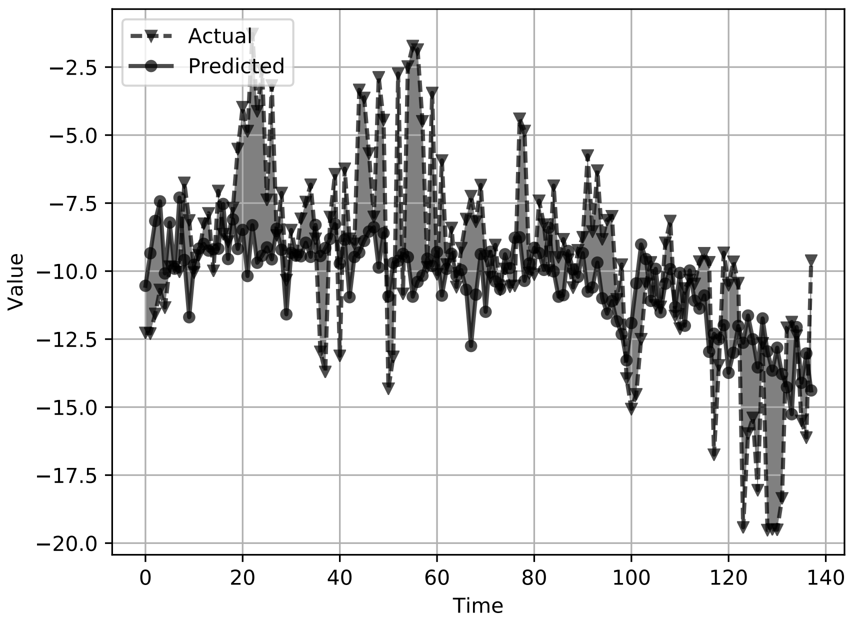
Using neural networks to simulate a functional relationship between radiometric readings and wind vectors is a novel application. Eventually, using techniques like this in a broader application to EWP, the computational burden would be shifted to something that is more adaptable to real-time global predictions. This is because all of the function simulations would be pre-trained into neural networks ready for application. This would be done in lieu of having to perform just-in-time calculations of the Primitive Equations of NWP. The trade-off would be the necessity to maintain a corpus of application-ready networks and train them in the background.

Remote sensing data is different from what is normally termed as “big data,” since remote sensing data is highly structured. However, data collection in this area results in enormous volumes that must be correlated spatially and temporally. Because of this, to use remote sensing data effectively in machine learning (i.e., in creating systems that are relatable and queryable such that timely predictions can be made), different data deployment regimes than those traditionally used in the meteorology community must be exploited. We intend to experiment with large scale distributed storage systems to facilitate an environment more favorable for this type of analysis. Such a system would be necessary to make any of this analysis usable for real-world data.

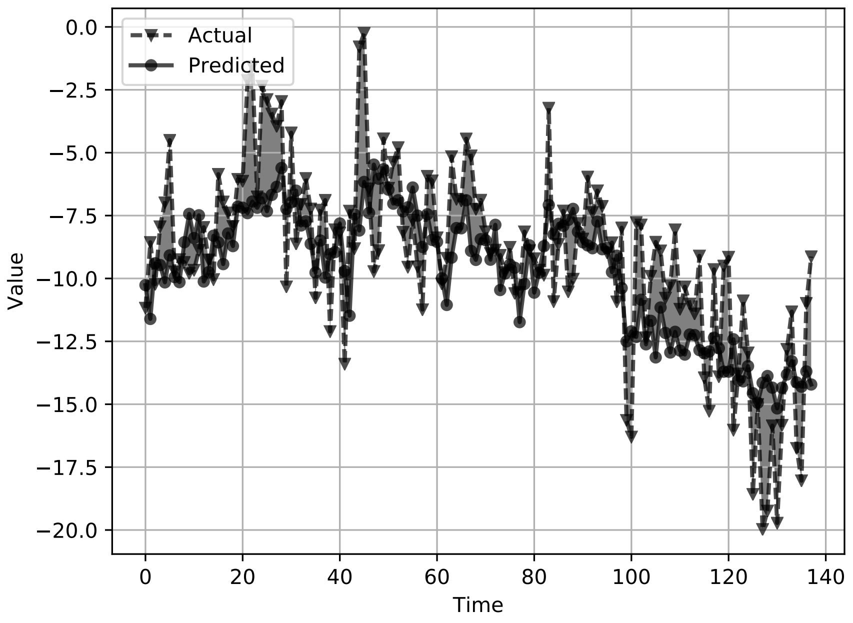
An avenue of research that will closely follow this paper is testing the longer-term (on the scale of hours) prediction capabilities of this procedure. This will involve generating models for all involved dimensions and so will require considerably more time for processing. The process will be to train the networks on the first part of the day, extrapolate for the next several hours, and test against the actual (WRF-simulated) data at the end of the day. Considerations will have to be made for the effects of the diurnal cycle on the behavior of the data. Recurrent neural networks may play a role here because they are specifically designed to address temporal data. Eventually, they will be able to account for n-order temporal consequences of some of the cyclical effects of the diurnal cycle, adding the ability for the model to automatically account for them.

The fact that we are inferring wind vector begs the question: can we create an accurate simulation environment using wind vector values that we predict (essentially using newly predicted wind vector values as inputs)? How far into the future can we run these predictions and still expect reasonable accuracies from the models? In doing this we would have to compare this deep-learning prediction framework with current states of the art in NWP (such as WRF) and see which system ends up with values that are closer to reality. A system like this would have entirely different computing burdens based on maintaining a corpus of neural networks, keeping them trained and swapping them in and out of use as their accuracy fluctuates. However, use of these systems would enable predictive models that run closer to real-time, adding insight for NWP practitioners and perhaps, one day, more accurate weather predictions.

There is an opportunity to extend what we have done here to a long-term prediction system. This system would be able both to make predictions about the conditions of the input variables and to use those predictions to predict the values of the response variable as well. Doing this would require a change in the architecture, as it would now be a multitask learning system that produces both predictions of the response variable and estimates about the conditions of the input variables in the next time step. A similar paradigm was explored by Li, et al., who approached large-scale automated prediction using cellular automata to predict precipitation [53]. However, they were using newly received radar data at each time step, not predicting the conditions of the input variables and allowing the system to run independently.

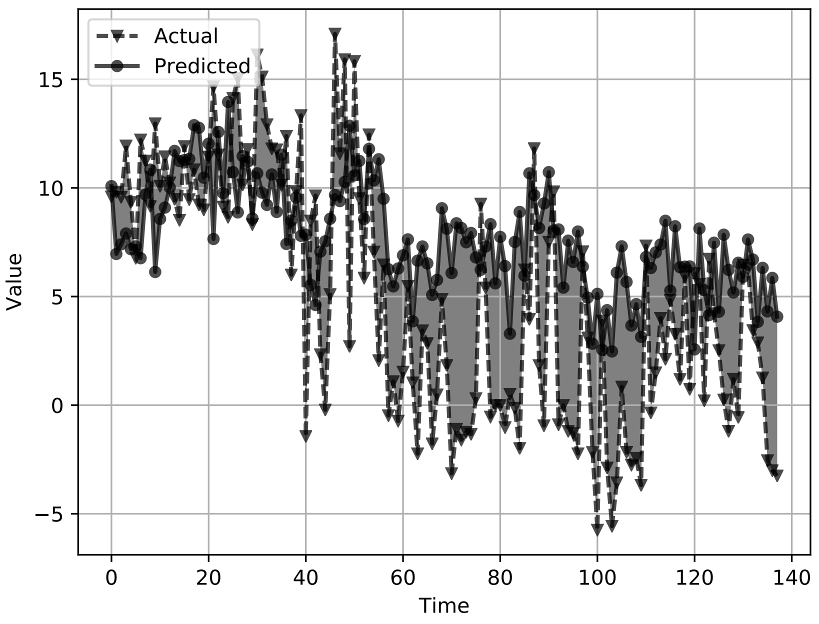


(a) u Component without Wind

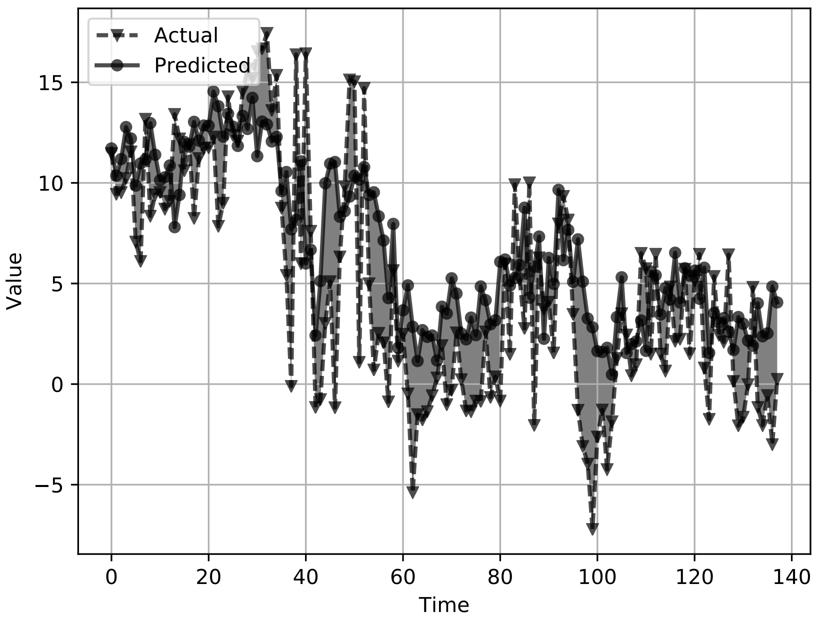


(b) u Component with Wind

Figure 3.12: Results from location 1: u component

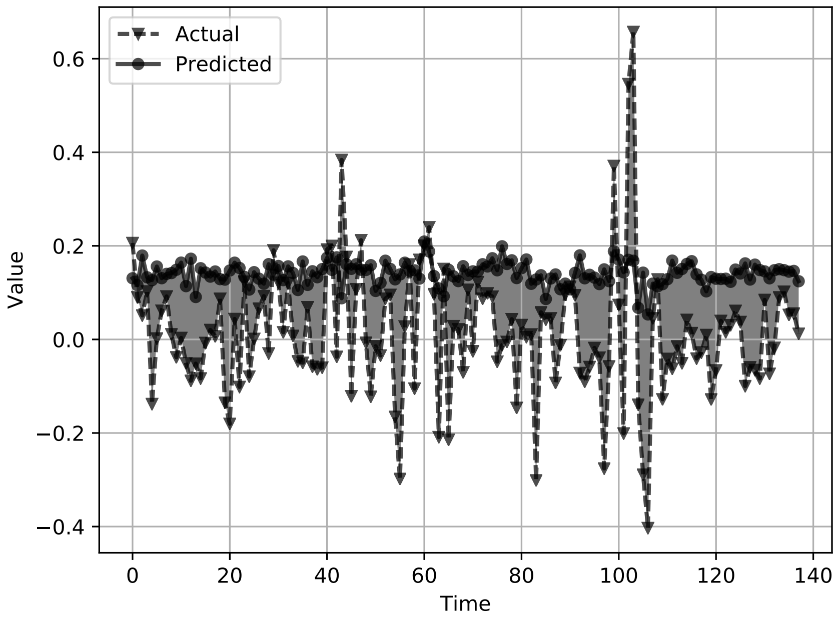


(a) v Component without Wind

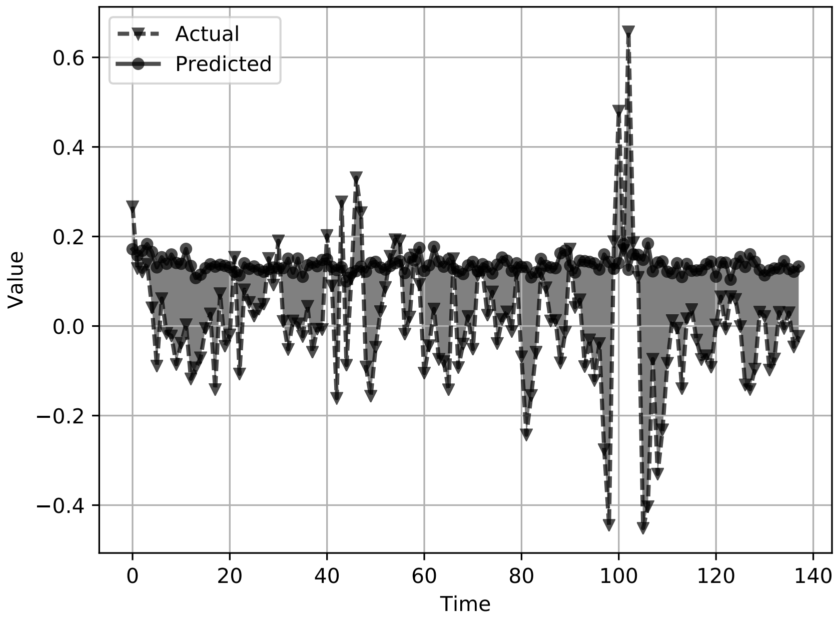


(b) v Component with Wind

Figure 3.13: Results from location 1: v component

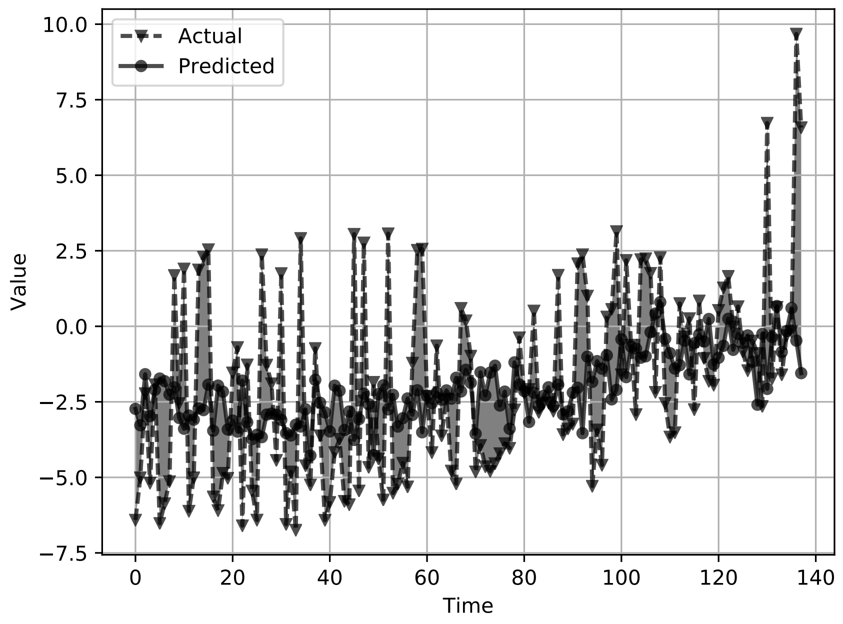


(a) w Component without Wind

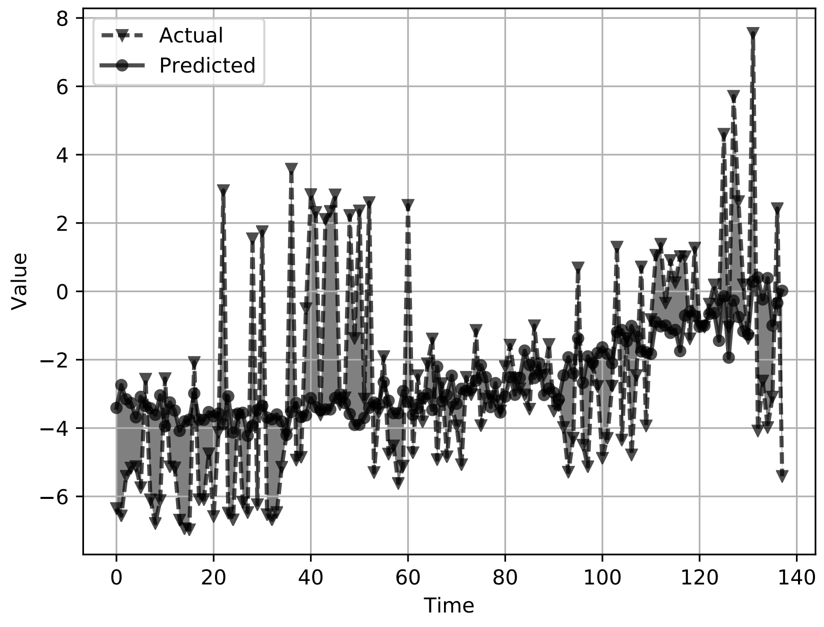


(b) w Component with Wind

Figure 3.14: Results from location 1: w component

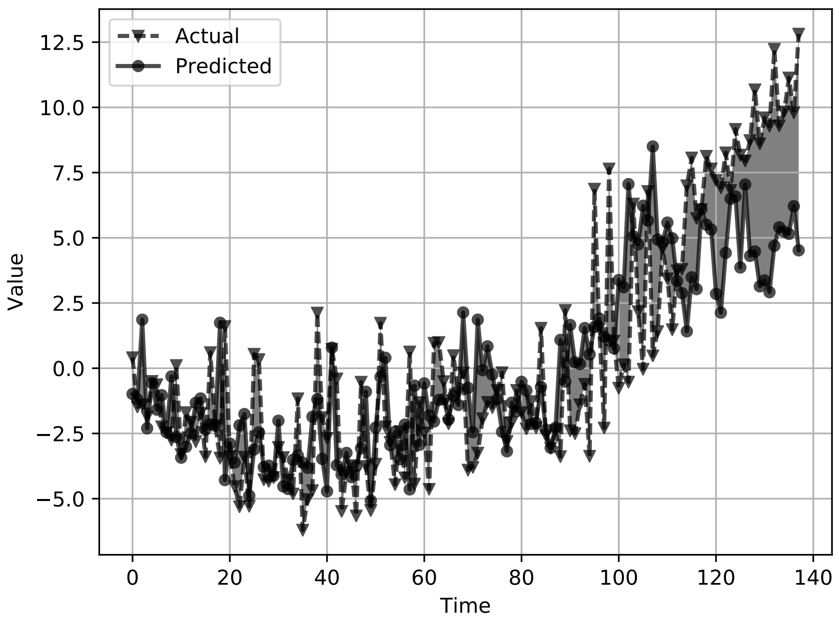


(a) u Component without Wind

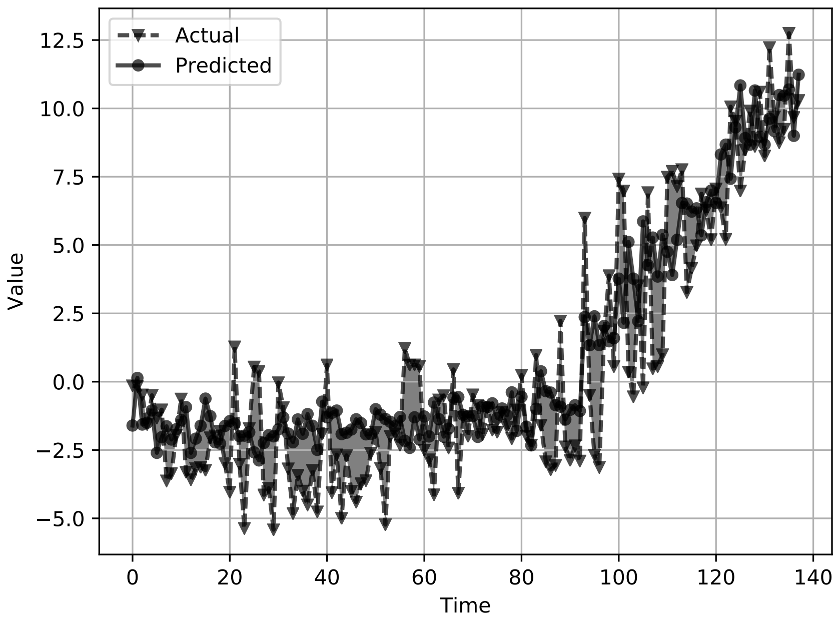


(b) u Component with Wind

Figure 3.15: Results from location 2: u component

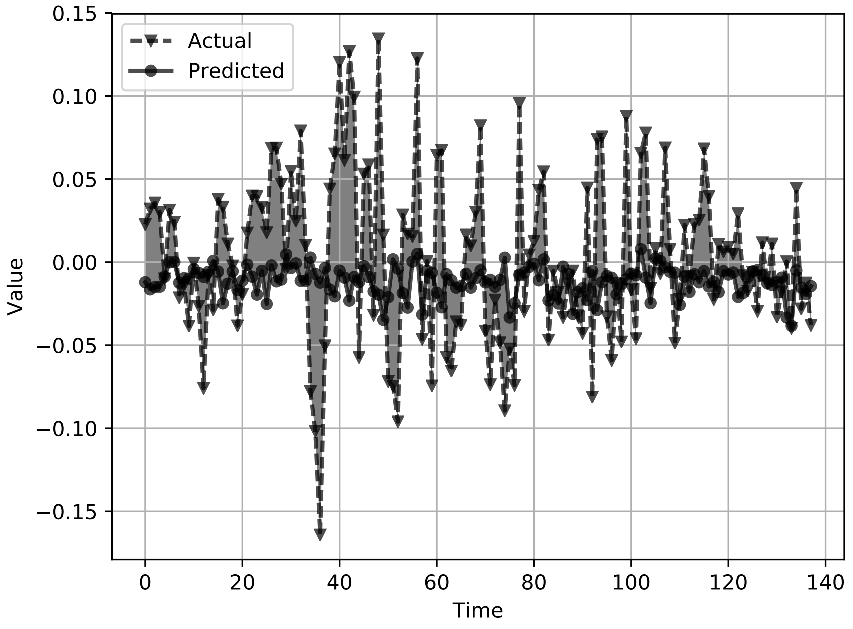


(a) v Component without Wind

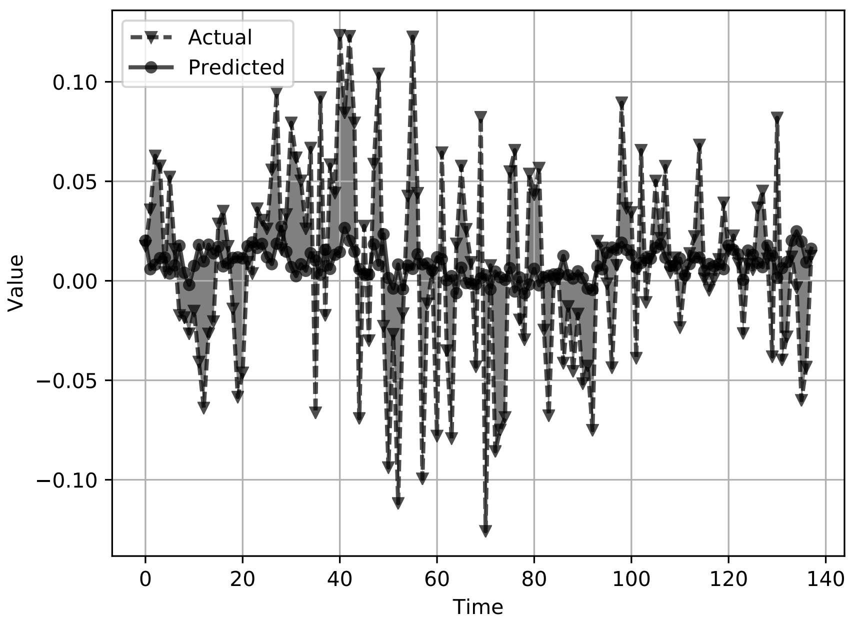


(b) v Component with Wind

Figure 3.16: Results from location 2: v component

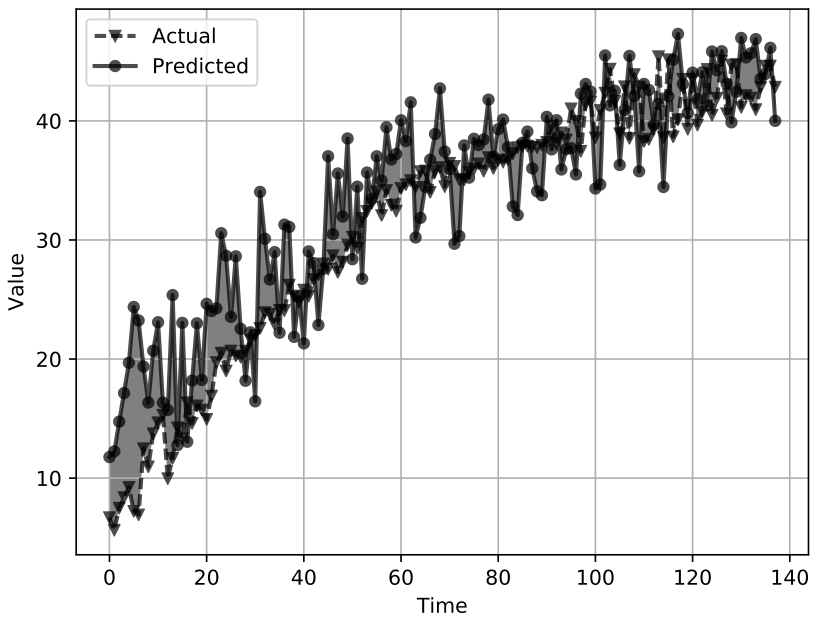


(a) w Component without Wind

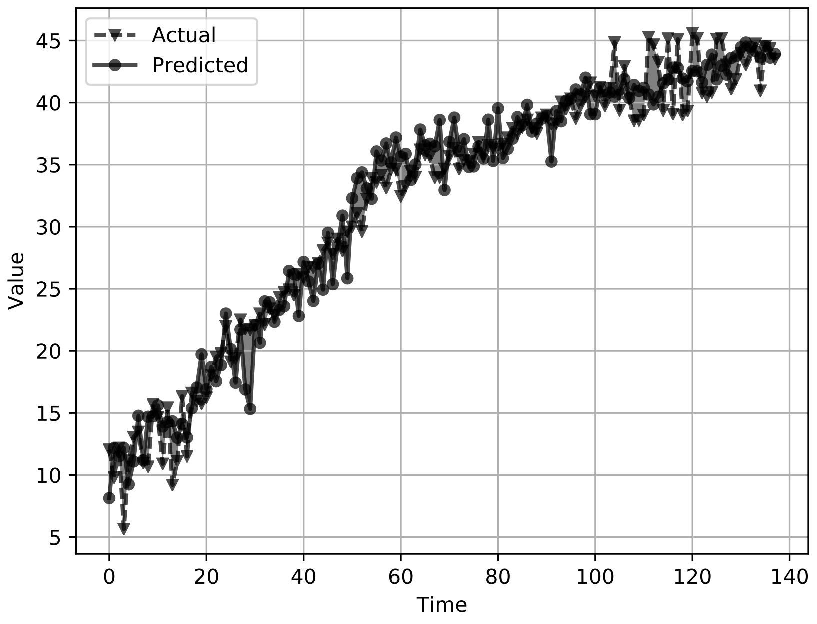


(b) w Component with Wind

Figure 3.17: Results from location 2: w component

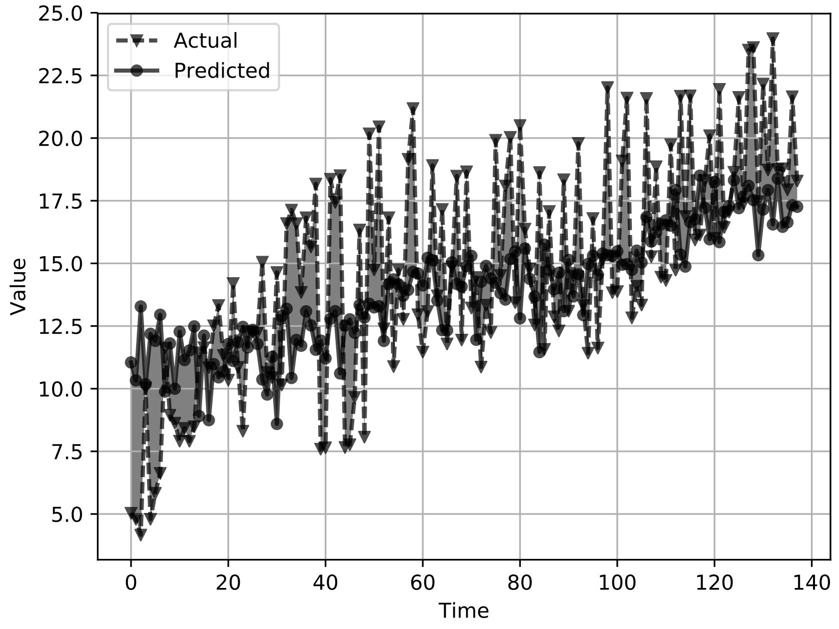


(a) u Component without Wind

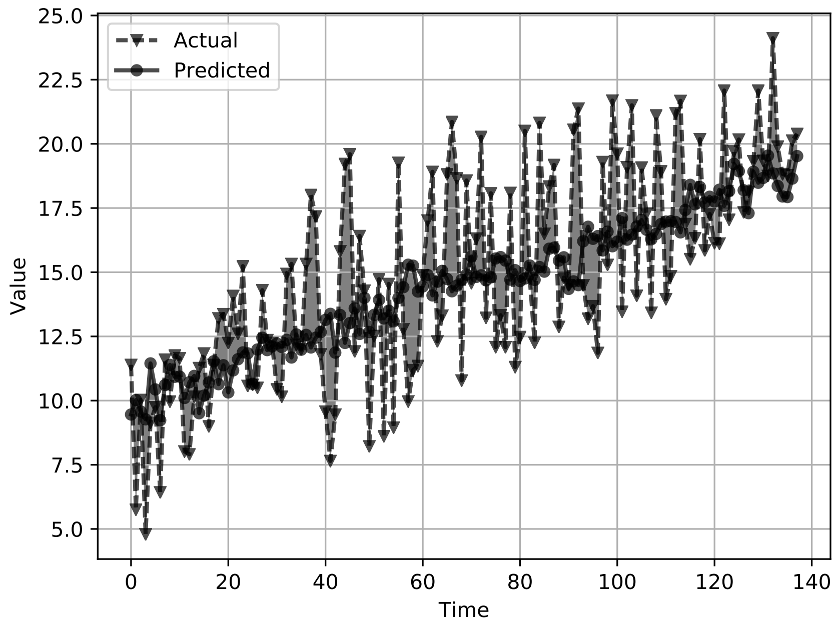


(b) u Component with Wind

Figure 3.18: Results from location 3: u component

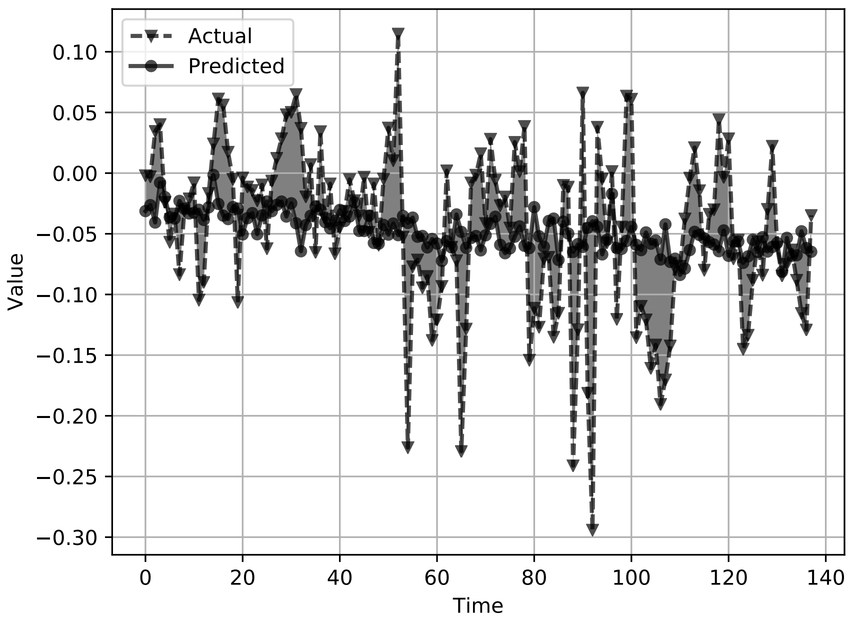


(a) v Component without Wind

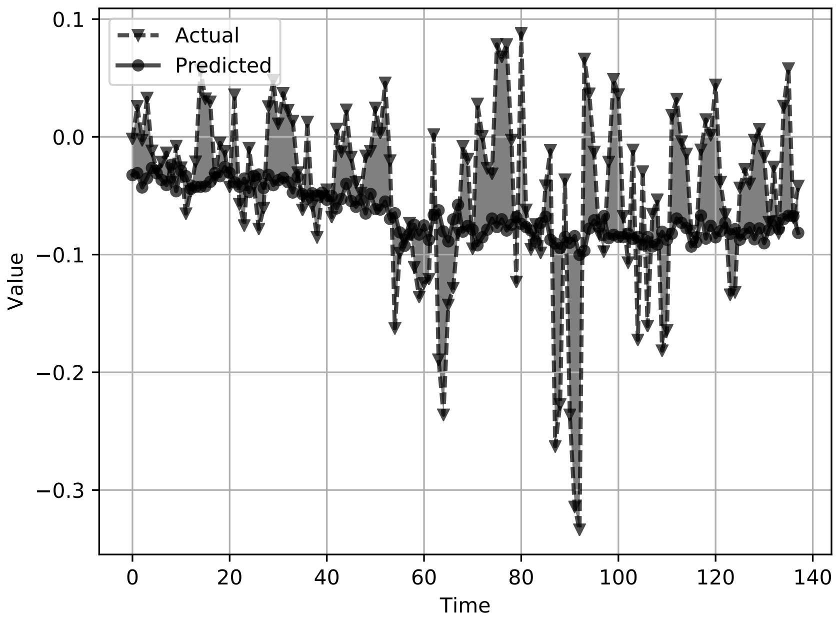


(b) v Component with Wind

Figure 3.19: Results from location 3: v component

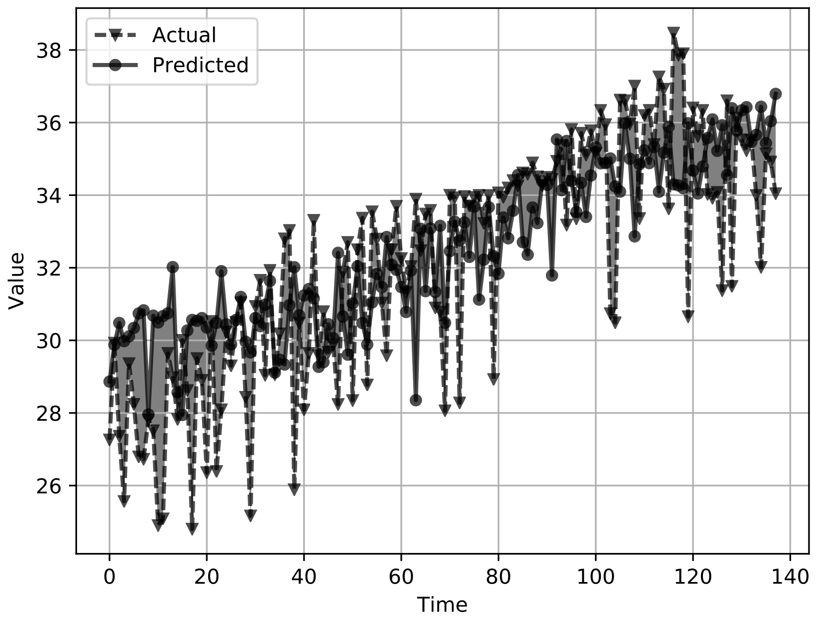


(a) w Component without Wind

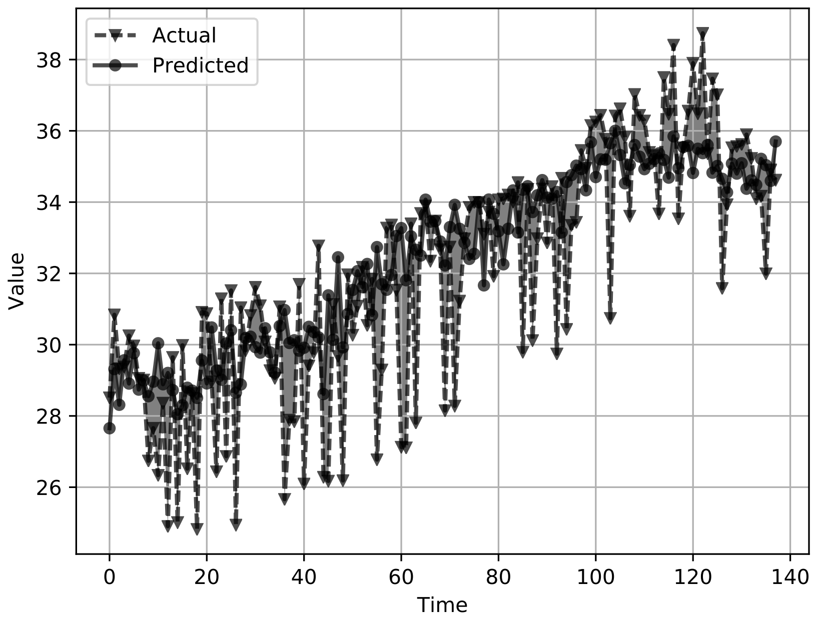


(b) w Component with Wind

Figure 3.20: Results from location 3: w component

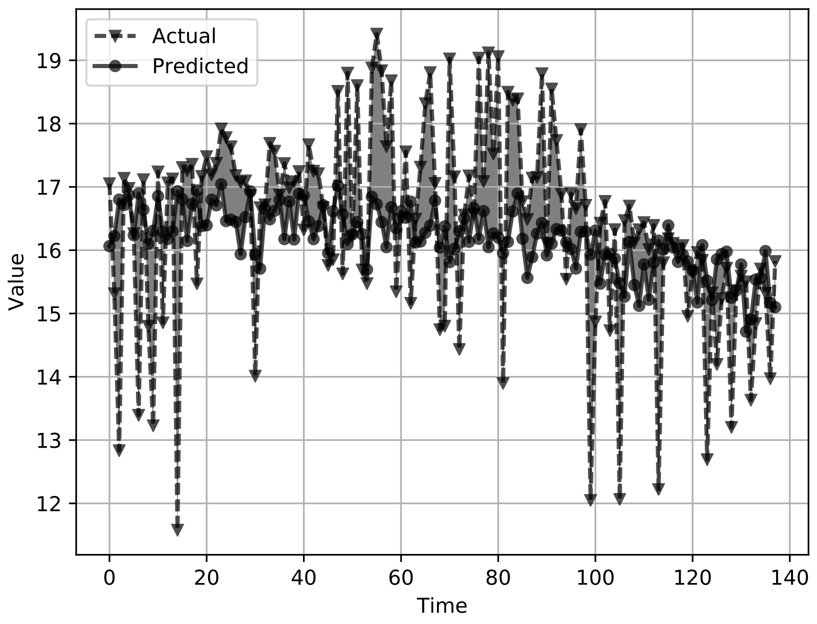


(a) u Component without Wind

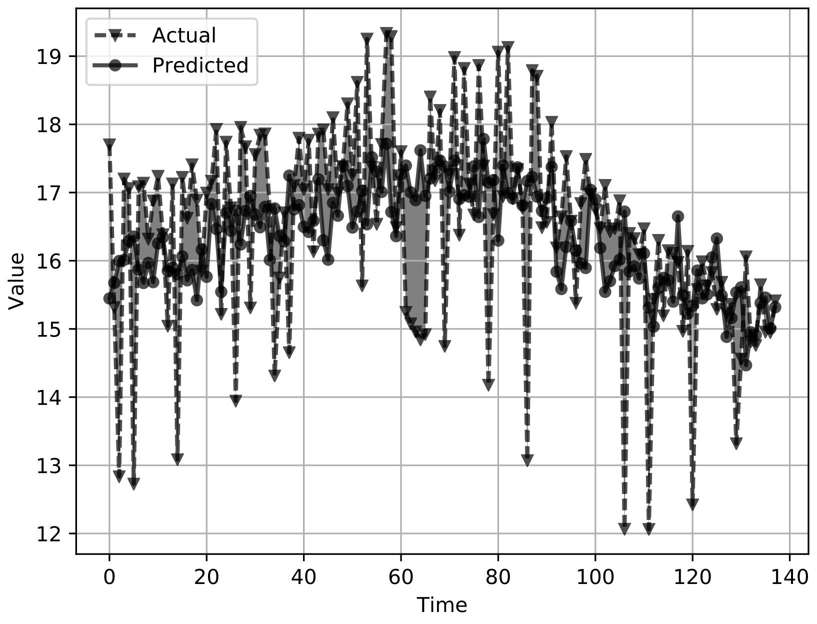


(b) u Component with Wind

Figure 3.21: Results from location 4: u component

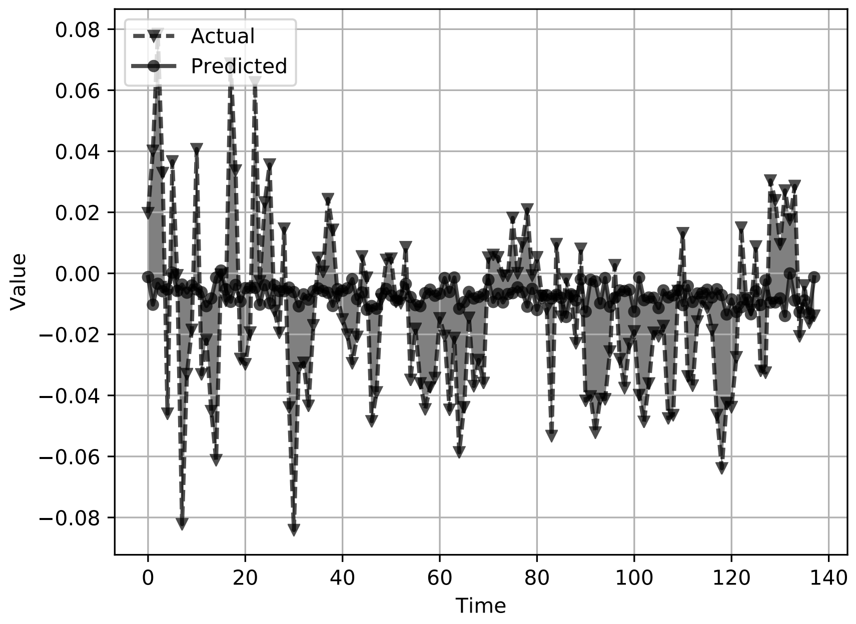


(a) v Component without Wind

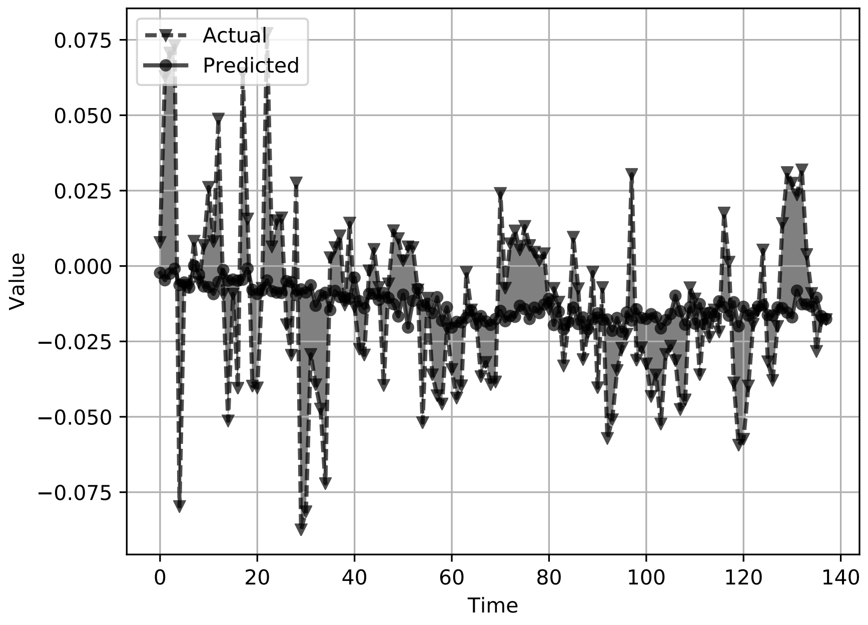


(b) v Component with Wind

Figure 3.22: Results from location 4: v component



(a) w Component without Wind



(b) w Component with Wind

Figure 3.23: Results from location 4: w component

CHAPTER FOUR

SPATIAL GENERALIZATION

4.1 Introduction

From a human perspective, the major factors that influence weather conditions in one area of the Earth's atmosphere and those that influence weather conditions in another seem to be the same. However, considering that the atmosphere is fluid, that conditions vary in time, that we have no access to the initial conditions of this fluid, and that we cannot enumerate the myriad forces that compose these factors, prediction is a very complex task. The simulation of functions that can produce predictions in this domain is a task for machine learning. If we assume that consistency holds between disparate locations around the Earth, transfer learning can be used to reduce the complexity of whole-Earth models.

In this chapter we begin to move knowledge gained in one area of the dataspace to other distinct areas of the dataspace where there are consistencies, but also some important differences. Here we are considering separate, discretized areas of the Earth's atmosphere to be these disparate settings. Some consistencies that exist among these diverse settings are that they are all composed of atmospheric gases, they are influenced by their own conditions and by the conditions of the areas that immediately surround them, and that this is not an entirely closed system. The differences are in that they have differing initial conditions and that they are affected by external forces in complex ways. For example, one area of the atmosphere may be over mountainous land and another may be over water, thus radically changing the associated weather dynamics. Further, one area may receive more solar radiation than another during the day.

Given such variation, one may be inclined to think that each area of the Earth would need to train its own model, resulting in a whole-Earth system that would be required to maintain millions of models (depending on the spatial resolution of each model). We seek a method whereby a fraction of these models would be required. In this paper we discuss experiments testing the generalization ability of representations learned by artificial neural networks (ANN) over a primary area of the Earth to areas adjacent to this primary area. We demonstrate that the functions learned to approximate wind vector components using radiometric data can be used to make determinations about the general trend of the wind vectors in these areas.

The results in this paper extend our work that we described in Chapter 3, demonstrating the extent to which trained networks can generalize to nearby regions in space. In the next section we will provide background on the problem domain. Next we will discuss the data that we are using and motivate how this study will support research into transfer learning. We will then discuss our approach. Our experimental design, execution, and results will then be explained. We will then discuss the broader implications of our results. Finally, we discuss our conclusions and the next steps for this research.

4.2 Background

4.2.1 Related Work

4.2.1.1 Stacked Autoencoders: In this work we use stacked autoencoders [9]. Autoencoders are used to create compact, reconstructible representations of data at a lower dimensionality. These representations can be stacked, using the outputs of hidden layers of one Autoencoder to encode deeper levels in a process called unsupervised pre-training (UPT). In this way, we can capture hierarchical internal representations of (theoretically) the most important features of the data [26]. This is discussed in more detail in Section 2.4.

4.2.1.2 Deep Learning in Weather Parameter Prediction: Research using deep neural networks for weather forecasting has been increasing over the past few years. This is partially in response to the advancement in deep belief network (DBN) training efficiency that was achieved by Hinton, *et al.* [43]. Since the structure of DBN's and stacked autoencoders are very similar, the training efficiency of the latter model was enhanced as well [38]. This is motivated by how important weather prediction is becoming to many industries. As a domain-specific example, Singh [91] used deep networks to determine how much wind energy was expected given the wind speed, humidity, and generation time.

Earlier, Dalto, Matusko, and Vasak [24] used deep networks for ultra-short-term (3 hour horizon with 10 minute resolution) wind forecasting. Their study demonstrated that deep neural networks benefit from an intelligent reduction in the number of input variables, allowing network training to complete in a reasonable time. They used Partial Mutual Information-based Input Variable Selection, which is a technique for determining how much redundancy is captured in the input variables, and selected variables that contributed the most information required to predict the response variables. Similar to our approach, they used separate networks for orthogonal wind vector components, not considering the vertical dimension at all. They used pre-trained stacked denoising autoencoders as their network model.

Narejo and Pasero [63] performed "meteonowcasting," (i.e., predicting conditions of the present or very near future) using multilayer pre-trained restricted Boltzmann machines. They used a method based on mutual information to determine the relevancy of atmospheric parameters at the prior time step to the current or future prediction of these parameters. The previous temperature and time of day were found to be useful for determining the temperature in the current time or near future. Each of these atmospheric parameters was treated separately, and the collection of parameters and topology of the networks used differed with each parameter they were trying to predict.

4.2.1.3 Transfer Learning: Transfer learning is the act of enabling a learner to use information from a model that was trained in one domain as a way to bootstrap training in another, related domain [67, 104]. One of the motivations for transfer learning is that data may not be available for a domain for which one wishes to make predictions. So the idea is to train on a domain in which there is plentiful data that is somehow related to the domain of interest and then fine tune with data that is available in the new domain. We assert that another motivation for using transfer learning is when there exists an extremely large number of domains that differ in subtle but important ways. Such a set of domains could be manifest as many geographic points on the globe that each have different forces influencing the conditions in their respective areas.

Hu, Zhang, and Zhou [44] used transfer learning for short-term wind speed prediction. They were addressing the problem where data for wind speeds over new wind farms was unavailable. The authors trained deep Autoencoders in areas that varied with respect to terrain, weather, and topography and still were able to produce models that learned abstractions that predicted wind speed in new farm areas effectively.

4.2.2 Problem Strategy

In our experiments, we attempt to determine current wind vector values using only radiometric data. Radiometers are unable to measure wind directly, but we assert that radiometric readings still provide information about wind vectors that can be used to train associated predictive models. The wind vector data we use comes from a separate wind-oriented model that was enriched using WRF and was taken over the same region at the same spatial and temporal resolution as the radiometric data we use for training. The dataset is explained in greater depth in Section 4.2.3. This wind vector data is used as ground truth for our models.

In predicting wind vectors, we want to know the extent to which the knowledge encoded in a neural network trained in one geographical area can be leveraged in making accurate predictions about conditions in another area. In each location around the Earth, weather conditions are all described by the same number and type of dimensions. This means that we are ensured of the ability to describe different locations using the same parameters. This is why we hypothesize the ability to transfer knowledge geographically; the consistency in parameters for each area allows us to feed the trained model with data from different regions.

4.2.3 Data

As was done in Chapter 3 we used a dataset created by [109] using an NWP simulator called the Weather Research and Forecasting Model (WRF). This dataset was created to support developing models from higher spatial and temporal resolutions than are currently available. The simulation used measurements of the East Coast of the United States during 24 hours of Hurricane Sandy in 2012. Specifically, the data is within an area bounded in the southwest at 26.4902°N, 81.6064°E and in the northeast at 41.2117°N, 60.3809°E (Figure 3.10). Actual radiometric and wind

Table 4.1: Latitude and longitude of the locations of interest

Name	**Latitude**	**Longitude**
Location 1	37.07	-73.79
Location 2	38.34	-62.63
Location 3	30.69	-75.65
Location 4	28.14	-64.49

measurements were taken at a far lower spatial and temporal resolution during this storm. WRF was used to interpolate measurements between these base measurements so the simulated dataset has a spatial resolution of 5km and a temporal resolution of 15 minutes.

For our study, as in Chapter 3, we focused on the four points shown in Figure 3.10a. Location 1 is an area that was in the eye of the storm for the duration of the simulation. Location 2 was in an area that was relatively unaffected. Strong rain bands were occurring in Location 3, and Location 4 had already been hit by the storm. Table 4.1 shows the latitudes and longitudes of these locations.

The data was provided in the form of *point clouds*, which are collections of points that exist in some spatial coordinate system. To make deep learning possible, we needed to bin these points spatially into discrete cells so that each data point in one cell would be representative of the conditions in that cell at that time. We used a Discrete Global Gridding System (DGGS) as the spatial binning mechanism [82]. Each data point in the point clouds was given a GIS Point designation on the globe, which allowed the data to be discretized into the DGGS using spatial queries. Each DGGS cell has 15km resolution in our model.

Radiometers are capable of obtaining vertical soundings of the atmospheric parameters. Therefore, our simulation included 60 vertical levels of readings, which we discretized into 10 levels.

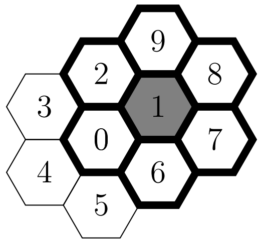


Figure 4.1: 7 cells representing one DGGS level

4.3 Approach

4.3.1 Cell Lattices

Figure 3.4 depicts our view of the space surrounding a given center cell for which we want to predict the wind vector components. This 21 cell lattice includes the 7 cells above the center cell, the 6 cells surrounding the center cell on the same level, and the 7 cells below the center cell. When training, we used measurements for representative points from all of these cells in the previous time slice (including wind vector information), as well as the wind vector components from the current time slice for the center cell as ground truth. In a second experiment, we just use radiometric data (no wind vectors) from the previous time slice.

The focus of this study is on determining whether we can transfer the knowledge from a model trained on one DGGS cell to a neighboring cell. This situation is depicted in Figure 4.1, which limits the process in one layer[1]. The idea is that we have trained a network for cell 0 using the information captured from the cell 0 and its surrounding cells. We then shift to a neighboring cell, such as cell 1 (in gray). For this cell, we use the network trained for cell 0 together with the measurements from

[1]Even though we have limited the presentation to one layer, the actual experiments considered the measurements over all three layers surrounding the target cell.

the previous time slice in cells surrounding cell 1 (outlined in bold). We repeat this process, determining wind vector components in cells 2 through 6 as well.

We call the collection of all of these cells, along with their data, the *super lattice* for cell 0. Thus, the super lattice includes all of the cells surrounding cell 0, which we call the *center lattice*, as well as all of the cells surrounding cells 1 through 6 as center cells. Working in the space of the super lattice provides a sense of the ability of the network trained for cell 0 to generalize to the neighboring cells, thus indicating a degree to which the trained network can be used in for transfer learning.

Our deep learning architecture consists of a four-layer stacked autoencoder, trained using greedy unsupervised pre-training. We used the same architecture and parameters discussed in Chapter 3 in these experiments.

4.4 Experiments

The intent of the experiments that we describe in this section is to determine the generalization capabilities of stacked autoencoders trained on specific areas of the dataspace to adjacent areas of the same dataspace. Though the differences in these areas are minimal, we get a sense of the flexibility of these trained models.

4.4.1 Overview

Our experiments proceeded using the following steps:

1. For a location of interest, instantiate the super lattice.

2. Pre-train the stacked autoencoder layers using the data from the center lattice.

3. Fine-tune the stacked autoencoder using the data from the center lattice.

4. Use the trained stacked autoencoder to predict the wind vector components for each of the peripheral cells.

Table 4.2: Features for each data point

Reading Source	Reading Name
Radiometry Measurements	Temperature
	Pressure
	Cloud Density
	Rain Density
	Ice Density
	Snow Density
	Graupel Density
Wind Speed	Wind u (East/West)
	Wind v (North/South)
	Wind w (Up/Down)

4.4.1.1 Super Lattice: Table 4.2 shows the data available for each cell in the super lattice. The top section of the table shows what measurements are available through passive microwave radiometry and the bottom portion shows what is available using other instruments (dropsondes, direct observation, etc.). We conducted experiments for input data that included the wind vector components for the previous time step for all of the cells in the super lattice, as well as input data that did not include these wind vector components. This approach allowed us to examine the ability of networks trained using only radiometric data to predict wind vectors and to assess the need for the wind vector data when moving to neighboring cells. The data that included the wind vectors from the previous time step had 231 dimensions and for the data without wind vectors had 168 dimensions.

4.4.1.2 Pre-Training and Fine Tuning: Our stacked autoencoders had 150, 140, 130, and 120 neurons per hidden layer when proceeding from input to output. As mentioned, the input layer consisted of 231 or 168 neurons depending on whether or not prior wind vectors were included. These layers were pre-trained using the training data from each cross validation fold. Following pre-training, the stacked autoencoder

Table 4.3: Average RMSE by level for the cells around location 3

	u Component		v Component		w Component	
Level	With Wind	Without Wind	With Wind	Without Wind	With Wind	Without Wind
High	97.345	65.290	16.704	7.917	0.003	0.002
Mid	0.492	0.345	0.205	0.162	0.001	0.001
Low	55.841	53.823	4.600	1.369	0.005	0.003

Table 4.4: Coefficient of determination (R^2) of deep networks for the mid level around location 3

Level	**Azimuth**	u Component		v Component		w Component	
		With Wind	Without Wind	With Wind	Without Wind	With Wind	Without Wind
Mid	Azimuth 1	0.989	0.993	0.984	0.985	0.817	0.658
	Azimuth 2	0.993	0.995	0.988	0.992	0.866	0.769
	Azimuth 3	0.989	0.992	0.984	0.988	0.823	0.736
	Azimuth 4	0.992	0.993	0.986	0.991	0.853	0.797
	Azimuth 5	0.993	0.995	0.987	0.991	0.726	0.627
	Azimuth 6	0.989	0.993	0.982	0.983	0.810	0.729

layers were fine-tuned using backpropagation. Each wind vector component (u, v, and w) was trained with a separate network as we described in Section 3.4.5.

4.4.1.3 Generalization: Once we trained our networks on the center lattice, we used each network to predict the current values of the wind vectors using data from the 20 neighboring lattices, using the 20 peripheral cells as the center cells.

To the best of our knowledge, we are the only ones using this data in this manner. Therefore, we are unable to compare our results directly to similar methods reported in the literature. Therefore, our comparison focused more on the potential dependence on prior wind vector information in generalizing predictive power in different geographic regions.

4.4.2 Results

We used the root mean squared error (RMSE) as our performance measure, which was the same performance measure used in Chapter 3. Table 4.3 shows the average RMSE of predictions for all three vector components for all cells surrounding the center cell (six surrounding azimuths above and below) for Location 3. This location is interesting because it lies directly in the rain band area south of the eye of the hurricane. As an indication of how much the variation in the u, v, and w components can be explained by the predicted values, Table 4.4 shows the coefficient of determination values (R^2) for the six cells surrounding the center cell at the same elevation, where we calculated R^2 as follows:

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2}$$

where f_i is the ith predicted value from the neural net, y_i is the ith target (observed) value, and $\bar{y}$ is the mean target (observed) value over the range of prediction.

The results for the peripheral cell at azimuth 1 at the medium elevation for Location 3 in Figure 3.10 are represented in Figures 4.2, 4.3, and 4.4. The magnitude of error over the range of the predictions is shown by the shaded region between the plots. The first column of charts shows the results of the predictions made for the data that included wind vectors from the previous time slice, and those on the right did not include wind vectors from the previous time slice. The first row depicts predictions for the u component, the second for the v component, and the third for the w component. This location, azimuth, and relative elevation was chosen because it was representative of the results obtained over the entire dataspace of peripheral cells.

4.5 Discussion and Contributions

For the locations studied, we found that our stacked autoencoders produced approximations of the real wind vectors. This is advantageous to the meteorological community because, as yet, a numerical, physics-based procedure to relate these radiometric measurements to wind vector values does not exist [33]. This methodology has the potential to enhance the data offering of radiometers with wind vector determinations.

We intentionally chose cells that were adjacent to the training cell to allow us to deviate spatially from the training space in a gradual manner. This is based on an assumption of locality that we made with regard to atmospheric dynamics—the forces that cause the atmosphere to behave the way it does—differ more as one moves further away from the initial training point [23]. If this is true then the networks so trained will be unable to capture a general model of the atmosphere when trained on any single area; however, we may be able to identify conditions under which generalization and transfer would be feasible.

The network trained on the center cell performed best at the middle elevation. The conditions in the cells above and below the center cell were different, but we were generally able to approximate the shape of the functions of the wind vectors in those areas. The average RMSE for the cells in the layers above and below the level depicted in Figures 4.2, 4.3, and 4.4 were significantly higher than for the cells at the same level as the cell upon which we trained the network. We may infer from this that there is an altitude-based sensitivity to the predictive capabilities to networks trained in this manner, though this conclusion will have to be tested using much larger datasets.

In a system of neural networks that makes predictions over a large spatio-temporal data space, different usage scenarios will have diverse modeling options, temporal (real-time) requirements, and accuracy standards. For example, we study three such scenarios.

The first learns a single model (e.g., a neural network) to act as a predictor over the entire space for all time slices. In this case, the model would need to have all input data and all response variables (along with location information) funneled through the same procedure. Also, since the model is being asked to account for a broader range of phenomena, the model itself may have to be expanded in complexity (e.g., depth or in number of nodes per layer) thus running the risk of overfitting due to having insufficient data to train properly.

The second scenario uses one model (i.e., network) for each area of interest, considering that model as being "responsible" for that area and no other. The expressivity of the model becomes a factor when discussing the types of functions that will need to be learned. For example, expressivity characterizes how the structural properties of a neural network affect the functions it is able to represent [74]. The intuitive advantage here is that the networks each concentrate their predictive power on a localized area whose data is affected by a lesser variety of underlying causes than it would be if it had to concentrate on the entire dataspace. Training and deploying models in this scenario can be parallelized but require a large stable of models (one for each area of interest) that all have to be updated when new data are available. A further complication is determining which model to apply in which situation, given the fact the properties of the data modeled are not fixed either in time or space.

The third scenario uses a collection of models trained on a relatively localized set (or cluster) of locations. The number of networks required would be somewhere between that required for the first scenario and the second scenario. The process involves determining how the data behaves in localized areas so that the division of the space can be done properly. This, in fact, is the scenario employed in later chapters of this dissertation. The tradeoffs in these scenarios pertain to the complexity of the models, the number of models that must be maintained, and the accuracy of the entire system.

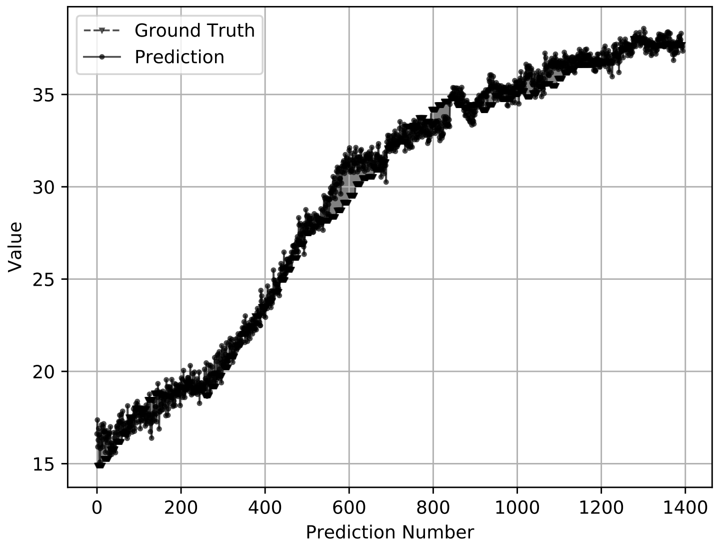
Our contribution in this chapter is that we produced models of the second scenario. We look to avoid both of the extremes while still maintaining a useful system for prediction. By determining the spatial extent of the effectiveness of our models, we are able to get an idea about how many models we will need to create a comprehensive set of models that are appropriate to the data under observation. In Chapter 6 we address the third scenario in constructing localized clusters of locations for training and application of neural networks.

4.6 Conclusions and Future Work

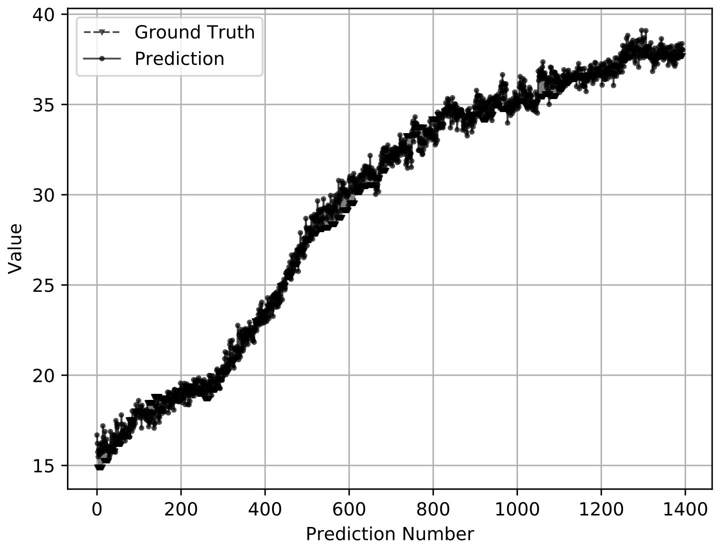
Based on the results of these experiments, we believe that we are able to apply deep learning with stacked autoencoders to capture important generalizations about the state of the atmosphere with respect to radiometric data and wind vectors. We have confirmed that these generalizations are capable of producing approximations for wind vector values in closely adjacent areas, thus suggesting the potential to be able to apply deep learning in NWP without the need for massive numbers of spatially distinct models. How versatile and transferable these generalizations are will be a subject of the next chapter, where we explore encodings and transferability in these networks.

We intend to focus the next steps of our research on determining what useful generalizations are captured when the networks are trained on one area and applied in a variety of novel situations. Among these situations will be locations that are further apart and the same locations in a different dataset depicting a different storm system (or no storm system). This investigation will also include an analysis of the effectiveness of pre-training on a primary region of interest and fine-tuning on the target region of interest of interest.

When our methodology is used with any compatible dataset, depending on the degree of local consistency that the dataset has, generalization may be used to a point with a tolerable level of error. After this point, if the data are sufficiently incompatible with the model with respect to dimensionality and spatial and temporal dependence, our method can determine what knowledge can be extracted from the trained models and transfer that knowledge to newly created models that address the unique characteristics of the target data. As future work, we will focus on exploring or developing approaches to determine the limits of the models in a given domain. We will use the results as a means to determine the number of models that will have to be trained to create a comprehensive system for prediction over a data space.

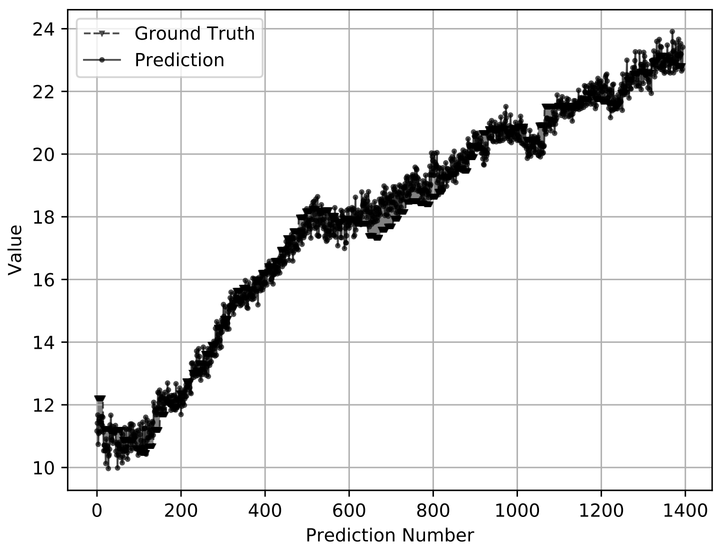


(a) u Component With Wind

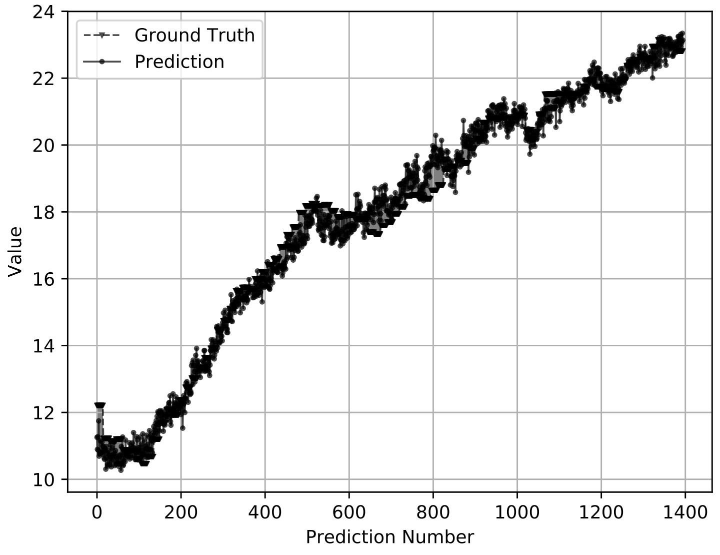


(b) u Component Without Wind

Figure 4.2: Predicted vs actual plots for the peripheral cell at azimuth 1 at the middle elevation for location 3, component u. The magnitude of error over the range of the predictions is shown by the shaded region between the plots.

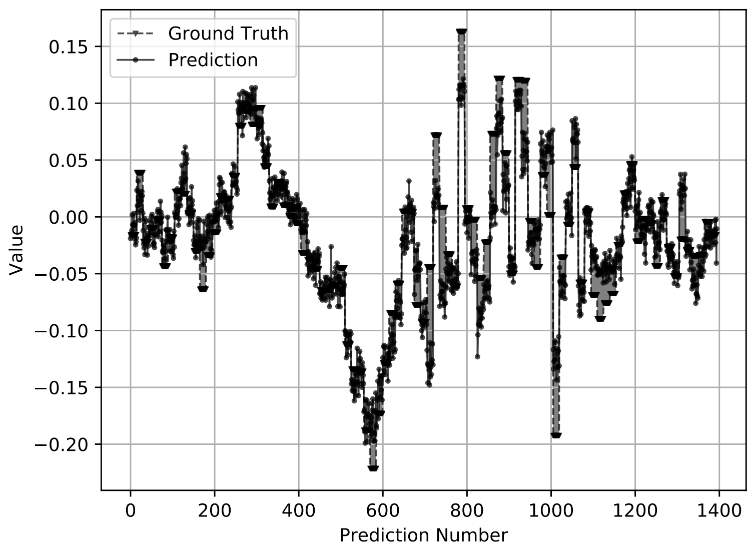


(a) v Component With Wind

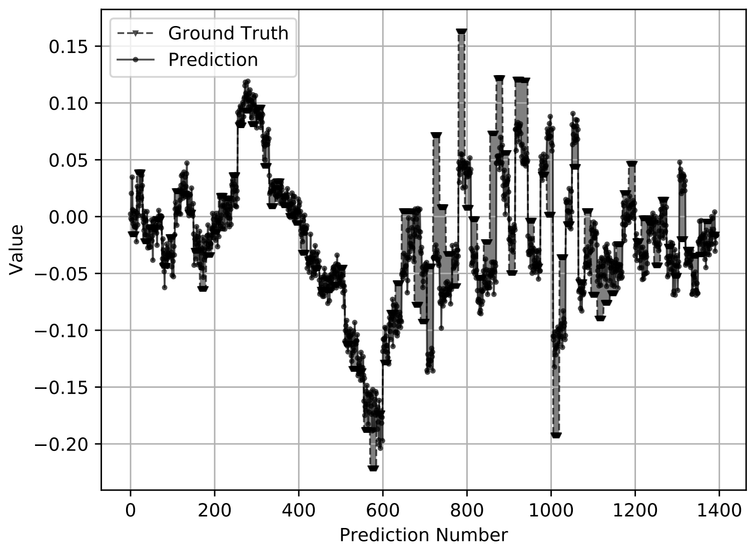


(b) v Component Without Wind

Figure 4.3: Predicted vs actual plots for the peripheral cell at azimuth 1 at the middle elevation for location 3, component v. The magnitude of error over the range of the predictions is shown by the shaded region between the plots.



(a) w Component With Wind



(b) w Component Without Wind

Figure 4.4: Predicted vs actual plots for the peripheral cell at azimuth 1 at the middle elevation for location 3, component w. The magnitude of error over the range of the predictions is shown by the shaded region between the plots.

CHAPTER FIVE

EXPLORING TRANSFERABILITY IN DEEP NETWORKS

5.1 Introduction

One of the the fundamental issues preventing the broad adoption of neural networks for many is the black-box aspect to them in that there is yet to be a comprehensive framework that allows an observer to understand and replicate what is going on inside the network. This predicament was noticed by DARPA, who issued a Broad Agency Announcement (BAA) for Explainable Artificial Intelligence in 2016 [1]. This BAA solicited techniques for training artificial neural networks (ANN) that would allow users to understand the reasoning that has been employed by the ANN.

It remains difficult to understand why ANNs produce the answers that they produce [85]. This opacity limits the ways that the ANN can assist in the understanding of the governing processes that underpin the system under examination. This has become a major area of research: to “look inside” ANNs and understand the “reasoning” that takes place. An advantage this understanding would confer is the ability to apply this data-derived logic in novel areas. Modern deep learning methods are effective and widely applicable to many critical problems, thus there is heavy motivation to use them even in the face of their opacity. If we cannot understand what is going on in these models maybe we can distill some of the knowledge inside of them so we can apply that as the data-derived logic.

A component of this issue is that there is a computational complexity in the training of these models. Combining this computational complexity with the amount and dimensionality of the data under investigation yields an exacerbated issue of effectiveness of the model per the time involved in creating it. There is much hope in figuring out how to transfer knowledge among models, whereby portions of trained models can be re-used as transplanted knowledge to act as intelligent starting points. Intuitively, this transferred knowledge, if more broadly applicable to the dataspace, will be knowledge that is more fundamental to the “understanding” of the abstract universe that encompasses both the source and the target of this transfer.

In this chapter we used a Stacked Autoencoder (SAE) on data that has been conditioned using techniques from functional data analysis (FDA). We then use spatial statistics to expose this transferable knowledge. After this we apply this insight to select internal components of the model learned to transfer them to models of other areas of the dataspace, and test the effectiveness of this transfer. To facilitate this, we created a highly controlled environment, bootstrapped by a single SAE with randomly initialized weights for the entire set of experiments. For each area of interest (AOI) we cloned this SAE and pre-trained the clones on the data from their respective AOIs in exactly the same fashion, meaning in the exact same order of instance precedence. This effectively removed all stochasticity in the training process, with the exception of the initial random weighting. It also allowed us to remove uncontrollable sources of variation and uncertainty that are normally endemic to the ANN training procedure and to concentrate on how different areas of the dataspace affect the aggregate training. This structure was the foundation for our experiments.

In meteorology and remote sensing domain, modeling and prediction has been a deterministic, mathematical process since the its beginning [101]. An advancement in ANN research in this specific area will provide an opportunity for deep learning to inform these traditional computational models, creating an exchange between data-derived machine learning techniques and mathematically derived statements about the fundamental dynamics governing the system.

This phase of our research contributes the following. Primarily, we provide a structured, controlled approach to evaluate the learnability and transferability of ANNs. To do this, we use functional data analysis and spatial statistics in a novel way to tightly control the experiment. Secondly, we develop an approach that applies the results of spatial analysis to make determinations of what subsets of neurons inside an ANN are transferable. Thirdly we test this transferability of these ANNs that have been trained in this manner.

5.2 Related Work

Yosinski, et al. studied transferable features in deep neural networks where the features were expressed as entire layers [cite “How Transferable ...”]. Later, Thompson, et al. extended this work, looking at how the layers held acoustic abstractions in language processing and how these abstractions would be transferable across languages [cite “How Transferable ... languages”]. We take a more granular approach in our work, asserting that feature abstractions do not necessarily include entire layers of the neural networks. If we are able to determine the subsets of nodes across these layers that encode valuable feature detectors, we should be able to create new networks that include more than one of these feature detectors. A benefit of this would be that the new networks may be able to produce accurate predictions in a wider variety of circumstances and overall training complexity would be reduced.

5.3 Data

As in the last section, the data we used for this phase of our research was the same data used in Chapter 3 and described in Section 3.2. In this phase we are concerned only with the fact that this data is highly multidimensional, spatial, temporal, and functional. The *multidimensionality* comes from the fact that, for each AOI, we have temperature, pressure, precipitation, humidity, and wind data. The *spatial* nature exists in that we are examining these same dimensions across a three-dimensional physical space, and these spatial relationships are a significant factor in the behavior of the system. It is *temporal* because data at each location are represented as a time series as it tracks a storm over a 24-hour period. It is also *functional* because changes in one area propagate through the space according to atmospheric forces and dynamics as a function of (among other things) space and time. A guiding principle in this research is that this data could be of any functional type, if what substitutes for the spatial and temporal perspectives still has distance and acts as the abscissa for the functional relationship among the data dimensions.

5.4 Functional Data Analysis

A prerequisite for the treatment of ANN training in the manner used in this chapter is that the data under study be functional in nature. Therefore, a critical assumption here is that the behavior of the data from each of the points of interest is influenced by the same common, global factors that influence all of these points together. Because of this, we hypothesize that the encodings that result from training networks on the data from each point of interest contain transferable knowledge. We assert that the understanding gained through this examination can broaden the predictive capability of ANNs trained through the procedure detailed herein that removes all unnecessary sources of stochasticity. We further hypothesize that spatio-temporally correlated feature detectors in a trained network can be extracted and used to train networks for other parts of the dataspace more efficiently and more accurately.

5.4.1 Description

When casting data in the functional data paradigm, data are analyzed as functions rather than as a collection of individual data points [50]. Viewing our meteorological data in this light is appropriate since the data vary smoothly as a function of time and distance. Also, since there is a hurricane moving through the space, each area of interest will undergo changes in data that track the hurricane moving in and out of the area of interest. The character of these changes will of course be appropriate to the position of the area of interest relative to the track of the hurricane.

5.4.2 A Simple Functional Data Analysis Example

Ramsay and Silverman [89] use the example from the Berkeley Growth Study [47] to explain functional data. In this work, the authors describe several children undergoing their pubertal growth spurt. Though the time, magnitude, and specific course of development may differ across the subjects, the process of maturation in this part of human development is generally similar in all subjects under study. This similarity is reflected in the similar transformation of the data as a function of time.

The following toy example illustrates the role that functional data analysis plays in this chapter. Suppose three people are standing along the side of a road as a firetruck passes by with its siren blaring. This is illustrated in Figure 5.1 with three pedestrians standing close to the road. When the three pedestrians are in the same relative position to the street as the firetruck passes, each of their experiences will be exactly the same in terms of the volume of the siren they perceive,

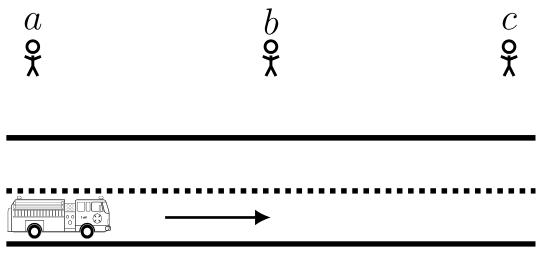


Figure 5.1: Pedestrians along road with passing firetruck

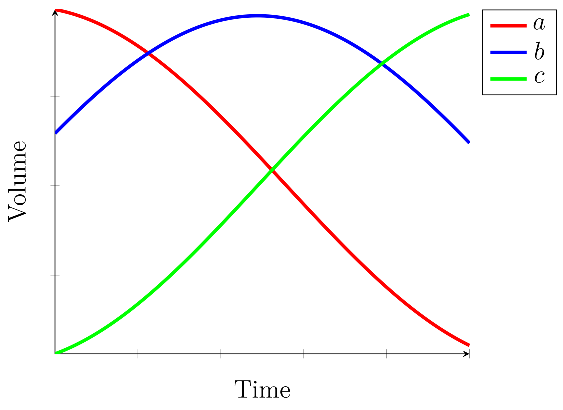


Figure 5.2: Volume levels by time for each pedestrian

albeit the experience will occur at different times. This is shown in the plot in Figure 5.2. However, when they are at different distances from the edge of the street and at inconsistent intervals along the length of the street, the power of the functional treatment of this data is much more readily apparent. Figure 5.3 shows three pedestrians positioned in this way, and Figure 5.4 shows a notional plot of the corresponding volume perceived by each pedestrian. Notice how added distance causes the overall volume to be lower and the volume change to be flatter, in contrast with the experience of the pedestrian closest to the street. Also, the time of experiencing the change in volume of the siren varies based on the lateral position of the pedestrian.

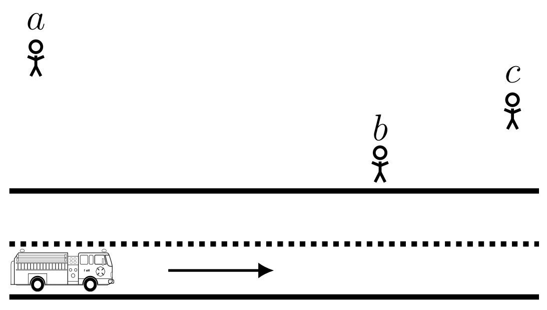


Figure 5.3: Pedestrians along road with passing firetruck

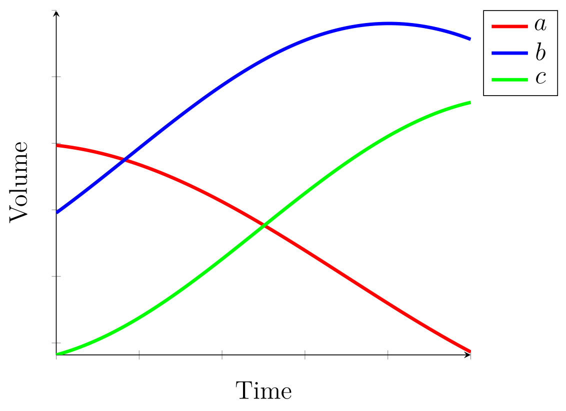


Figure 5.4: Volume levels by time for each pedestrian

Functional data registration (FDR) is a procedure that causes the peaks of these curves to become aligned (shift registration) and the amplitudes to be modified (amplitude registration) as much as possible to bring the features of the functions into alignment [89]. The objective is to minimize the impact of the differences in the dimensions upon which the function is dependent so that the features themselves can be contrasted. Hurricane Sandy, depicted in our example data, is analogous to the firetruck in our toy example. The Hurricane is a spatio-temporal phenomenon that is moving through our area of study. As the phenomenon makes its way through every location on the map, its “locations” in the phenomenon affect “locations” in the space in a related way, like the siren on the firetruck.

5.5 Approach

We performed a two-stage training of a set of SAEs to extract encoded information that we could use to transfer through the dataspace. The autoencoders that make up the hidden layers were trained using unsupervised pre-training. All sources of randomness in the model, and in the model’s training, were removed with the exception of the primary random initialization of the weights in the prototype network. Each network corresponding to each AOI in the dataspace was cloned from this prototype network, resulting in identical initial networks. Continuing the motif of removing all sources of randomness, both pre-training and fine tuning used data in the exact same temporal order for each AOI, resulting in a consistent time-correspondence across the dataspace.

This treatment allowed us to trace how the data affected a consistent model, which ensured a focus on the exploration of the dataspace rather than a concentration on optimizing and refining the model itself. The analysis proceeded on the weights and activations that resulted from differential interactions across the dataspace. This allowed us to use variograms from spatial statistics to determine what is and what is not shareable data that generalizes across the entire area.

5.5.1 Data Preparation

5.5.1.1 Area of Interest Instance Data

Figure 5.5 shows the specific subset of the areas of interest where our data was collected. The white figure in the background on the left side of the figure shows the Eastern seaboard of the United States from Long Island on the upper left to Florida and Grand Bahama in the lower left. The

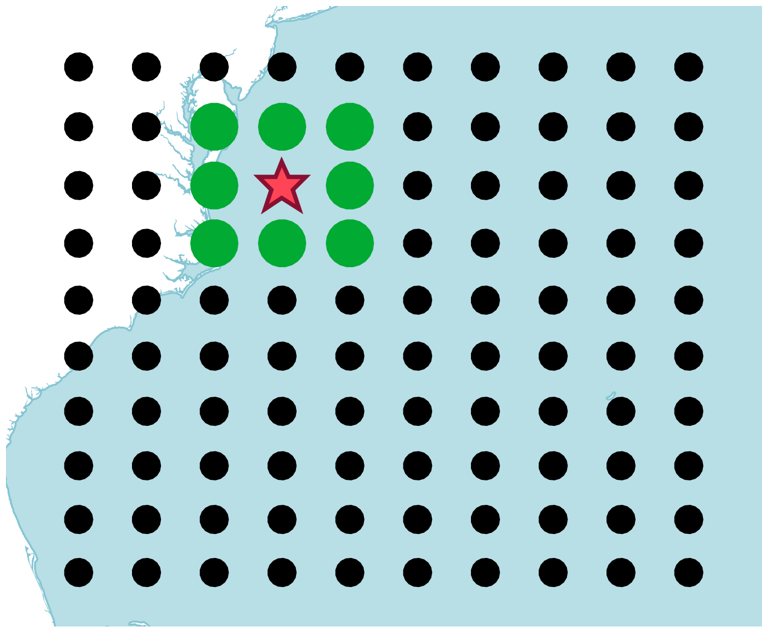


Figure 5.5: Analysis locations for training the neural networks

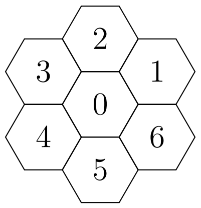


Figure 5.6: 7 cells representing one DGGS level (repeated here for convenience)

blue area depicts the Atlantic Ocean. The dots (including the green dots and the red star) show the locations of each geographical area we analyzed. Each dot represents the center of a grid cell in a geodesic Discrete Global Gridding System (DGGS) [82] superimposed over the entire planet.

The areas of interest were arranged as depicted in Figure 5.6, repeated here from Figure for convenience. Each numbered cell corresponds to a DGGS cell of 15km resolution. As in previous chapters, we concentrated on predicting the wind vector conditions in cell 0 at each location for each succeeding time step in the dataset. This is consistent with the previous assumptions we have made: that the wind vector values in cell 0 for the current time slice (excluding those forces not represented in the data) can be determined by the radiometric readings of that same cell for the current time slice and the radiometric and wind readings of all cells in the previous time slice.

5.5.1.2 Time Shift Instance Data Table 5.1 represents data instances as they are presented to the system, where t_{in} corresponds to the time that the input data was captured, t_{out} corresponds to the time the output data was captured, $\{t_0, \ldots, t_m\}$ are the specific time steps, $\{x_0, \ldots, x_n\}$ represent the input data variables, and $\{u, v, w\}$ represent the output data variables. These output are the variables that we are trying to predict, with u representing the East-West wind vector component, v representing the North-South component, and w representing the vertical component. In the Figure, the input variable $x_{i,j}$ represents an instance of the variable x_j at time slice i. The output variable u_1 represents an instance of the variable u at time slice 1 and the variable u_m represents an instance of the variable u at time slice m. The time variables are offset from each other by one time step, with t_{in} starting at time slice 0 and t_{out} starting at time slice 1. Thus we set this up as a one-step time series prediction problem.

To address the temporal nature of this data, we assume the conditions of each variable at each time step influence the conditions of the output variables at the next time step. With this in mind, to create our predictive model, we shift the time for the output variables, associating t_{i-1} for the input variables with t_i for the output variables.

Table 5.1: Time shifted data

$\boldsymbol{t_{in}}$	$\boldsymbol{x_0}$	$\boldsymbol{x_1}$	$\boldsymbol{\cdots}$	$\boldsymbol{x_n}$	$\boldsymbol{t_{out}}$	$\boldsymbol{u}$	$\boldsymbol{v}$	$\boldsymbol{w}$
$\boldsymbol{t_0}$	$x_{0,0}$	$x_{0,1}$	$\cdots$	$x_{0,n}$	$\boldsymbol{t_1}$	u_1	v_1	w_1
$\boldsymbol{t_1}$	$x_{1,0}$	$x_{1,1}$	$\cdots$	$x_{1,n}$	$\boldsymbol{t_2}$	u_2	v_2	w_2
$\boldsymbol{\vdots}$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\boldsymbol{\vdots}$	$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{t_{m-1}}$	$x_{m-1,0}$	$x_{m-1,1}$	$\cdots$	$x_{m-1,n}$	$\boldsymbol{t_m}$	u_m	v_m	w_m

5.5.1.3 Scale Data The scale of the input and output data variables differed widely. The minimum range of the unnormalized data variables was between 0.0 and 1.2713466e-09. The maximum range of the unnormalized data variables was between -50.939037 and 42.185276. Each of the data variables was also captured using differing units, for example: kilometers per hour for wind and degrees Celsius for temperature. To bring all of the data into a common scale, we scaled the values for each of the variables to a range between 0 and 1. Each of the data variables was scaled individually so that the functional character of the data represented was preserved. We retained each of the scalers so that the predictions could be compared to the values that they were supposed to simulate.

5.5.1.4 Separation of Input and Output Data We were interested in the functional relationship between atmospheric variables from radiometry readings and the wind vectors in the same area, so the radiometry measurements were the main features used for prediction. In the future we would like to use the wind vector predictions from the prior time step as inputs for the next time step, creating a more comprehensive system with enhanced predictive capabilities. But for now we wanted to provide the networks with as little information as possible about the state of the wind vector, save for the ground truth outputs in training.

5.5.1.5 Functional Data Registration To bring the shapes of each of the input dimensions into greater relief, rendering the data comparable, we performed functional data registration over the dataset. This included temporally shifting the functional form of the data to bring the features into alignment regarding time and intensity. This is called shift and amplitude registration—shift registration describing the adjustment of the function's time dimension to align features along the abscissa, and amplitude registration describing the increase or reduction in intensity to align features along the ordinate.

Functional treatment of the data required its transformation into functional form. We converted the data into a set of basis functions and a corresponding set of coefficients. The two choices for the basis functions explained in [89] are the Fourier basis and the spline basis. Since the Fourier basis function is primarily used in periodic data, and our data only spanned a single day, we chose the spline basis as being better suited for our non-periodic dataset. To be clear, we are not using splines as a model of the data. We are using them as a part of the procedure for the registration of the data.

A spline is defined as, "A piecewise polynomial function that can have a locally very simple form, yet at the same time be globally flexible and smooth [105]." The function components are joined smoothly at the breakpoints that separate the functions because of the constraint that the functions, their derivatives, and their second derivatives be equal at their junctions. The junctions in this case are the actual data points. To create the sample functions, for each function we define a system of spline basis functions, and since a scalar multiple of a spline function is still a spline function, represent them as linear combinations of these basis functions [89].

For shift registration of the curves, we are interested in the values:

$$x_i^*(t) = x_i(t + \delta_i)$$

where x_i is the sample function, t is the position (in our case the time) along the abscissa, and the shift parameters δ_i align the curves. We estimate the mean function $\hat{\mu}(t)$ over the interval in which we are interested. The mean is estimated using the spline smoothing criterion ($PenSSE$) [89]:

$$PenSSE = \sum_{j=1}^{n_i}(y_{ij} - x_i(t))^2 + \lambda \int \left(\frac{d^2x(t)}{dt^2}\right)^2 dt$$

where n is the number of observations, λ is a roughness penalty, and y_{ij} is fit by the curves of x_i [50]. We then minimize the following criterion with respect to δ_i:

$$\begin{aligned}\frac{\partial}{\partial \delta_i} REGSSE &= 2\int_T \{x_i(t+\delta_i) - \hat{\mu}(t)\}\left(\frac{dx_i(t)}{dt}\right) dt \\ \frac{\partial^2}{\partial \delta_i^2} REGSSE &= 2\int_T \{x_i(t+\delta_i) - \hat{\mu}(t)\}\left(\frac{d^2x_i(t)}{dt^2}\right) dt \\ &+ 2\int_T \left(\frac{dx_i(t)}{dt}\right)^2 dt\end{aligned}$$

where $REGSSE$ is the global registration criterion and T is the time interval in which we are interested. The process is iterated, re-computing the mean $\hat{\mu}(t)$ from the registered curves, and re-computing the shifts δ_i using the following [89]:

$$\delta_i^{(v)} = \delta_i^{(v-1)} - \alpha \frac{(\partial/\partial\delta_i)REGSSE}{(\partial^2/\partial\delta_i^2)REGSSE} \tag{5.1}$$

where v is the iteration number and α is a step-size parameter.

Algorithm 5.1 shows the shift registration procedure. The variables ξ and ζ are convergence parameters for the $(\partial/\partial\delta_i)REGSSE$ and $(\partial^2/\partial\delta_i^2)REGSSE$ functions. The parameters for this procedure correspond to the variables in the respective formulas. Line 2 shows that the shift parameter is initially estimated to be zero.

Algorithm 5.1 Shift registration [89]

```
1: procedure SHIFT_REG($x_i, y_{ij}, t, \alpha, \xi, \zeta$)
2:     $\delta_i = 0$
3:     $v = 0$                                        ▷ Iteration counter
4:     while $(\partial/\partial\delta_i)REGSSE > \xi$ and $(\partial^2/\partial\delta_i^2)REGSSE > \zeta$ do
5:         $\hat{\mu} = PenSSE$
6:         re-compute $\delta_i^{(v)}$ as in Equation 5.1
7:         $v = v + 1$
8:     end while
9:     return $\delta_i$
10: end procedure
```

The convergence criteria are shown in line 4. Lines 5 and 6 show the updating of the mean estimate and the re-computation of the shift parameter.

5.5.1.6 Train, Validate, and Test Separation Since we were exploring the dataspace rather than exhaustively validating a training methodology, it was important that we kept the treatment of the data consistent among training, validation, and testing sets. For this reason, the time indices for the training, validation, and testing segments of the data were pre-selected before any of these processes proceeded. After completing the aforementioned procedures, there were 93 data instances for each area of interest. Since there were 93 total time slices for the data, for training we reserved 73 of these, selected at random, and 10 each for validation and testing.

5.5.1.7 Sequential Data Training The data from all of the points of interest on the map are bound together by time, with the first instance for each location happening at the same time as the first instance in all other locations, and so on. Normally, during the training of neural networks, the examples are fed to the procedure randomly. This would have the effect of scrambling the temporal sequence across the dataspace, rendering the instances incomparable. Since we wanted total consistency in the training of these networks, and so that there was no randomization during the training process, exactly one ordering of data was used universally for all locations. This was done to ensure that the same time indices were used for all areas of interest and all of the training epochs corresponded with each other. This ordering was pre-selected and was applied system-wide.

5.5.2 Layer Pre-Training

Figure 5.7 shows the stages of the pre-training procedure and how the general process flow proceeds. The large rectangle in the center are the major parts that we contributed in this study. Each item in the diagram is explained in the following sections.

5.5.2.1 Prototype Network

In the interest of removing all sources of variation in the pre-training procedure, the networks trained on the data from each area of interest were cloned from a single, randomly initialized, autoencoder. Therefore, all pre-training had the same random starting point. The *prototype autoencoder* was a single layer of 150 nodes and its layer weights were each initialized to random values between -1 and $+1$.

5.5.2.2 Node Profiles and Pairwise Dot Products

After pre-training, each network was the result of the original, cloned autoencoder having been pre-trained on data from its respective area of interest, and therefore contained the compressed autoencoder representation of that area of the dataspace. This is crucial to this investigation because it is a goal to expose the different ways in which different areas of aggregate environment differently affects a homogeneous system in order to extract knowledge that can be used in other areas of the dataspace.

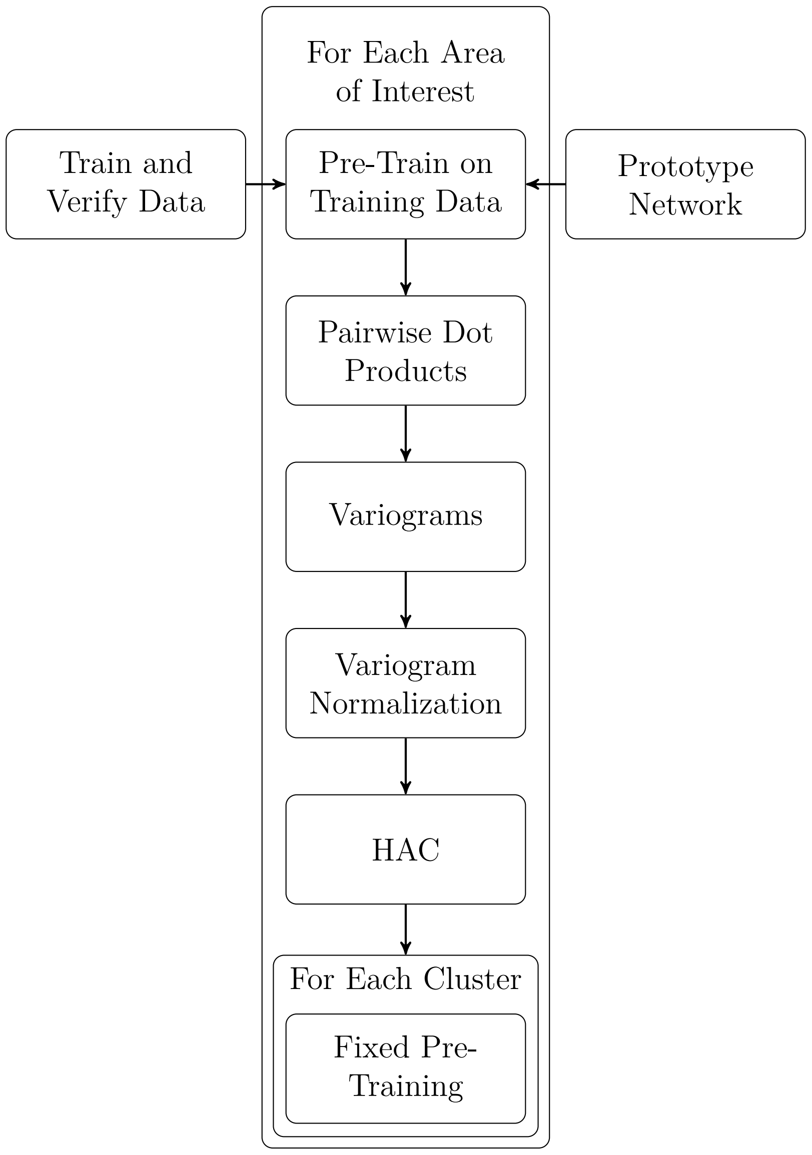


Figure 5.7: Pre-training

To compare what was learned by each node across the dataspace, we collected each node's incoming weight vector. Specifically, for each autoencoder and for each area of interest we collected the weight vector of each corresponding node and arranged them into a similarity matrix. Because of the way the networks were trained we assert that if their weight vectors are the same, the feature learned by

Table 5.2: Node profiles

(a) Node profiles

	$\boldsymbol{a_0}$	$\boldsymbol{a_1}$	$\cdots$	$\boldsymbol{a_c}$
$\boldsymbol{v_0}$	$w_{0,0}$	$w_{0,1}$		$w_{0,c}$
$\boldsymbol{v_1}$	$w_{1,0}$	$w_{1,1}$		
$\vdots$			$\ddots$	
$\boldsymbol{v_e}$	$w_{e,0}$			$w_{e,c}$

(b) Pairwise dot products

	$\boldsymbol{a_0}$	$\boldsymbol{a_1}$	$\cdots$	$\boldsymbol{a_c}$
$\boldsymbol{a_0}$	$p_{0,0}$	$p_{0,1}$		$p_{0,c}$
$\boldsymbol{a_1}$	$p_{1,0}$	$p_{1,1}$		
$\vdots$			$\ddots$	
$\boldsymbol{a_c}$	$p_{e,0}$			$p_{c,c}$

a particular node at one location corresponds to the same feature learned by that node in another location.

More formally, suppose we have two autoencoders A_1 and A_2. Suppose we order the hidden nodes of each autoencoder as $h_1, \ldots, h_n$. Because of the controlled pre-training procedure and the fact each autoencoder started from the same state, we assert that node $h_i^{A_1}$ and $h_i^{A_2}$ are examining the same feature of the underlying data and are, therefore, comparable. This is because we are using a controlled pre-training procedure that uses gradient descent. Since the autoencoders are being trained using gradient descent they will converge to a stable local optimum. We computed the pairwise dot products of the node weight vectors for each node as the measure of similarity between each of the node profiles for each location. Table 5.2 depicts the results of the node profile extraction proceure.

5.5.2.3 Variograms Having the matrix of pairwise dot products based on location allowed us to analyze the variance as a function of distance between each node. For this we used variograms [13, 35], which depict the differences in the dot products for all of the nodes for each location. This tool enabled us to examine the extent to which the results of unsupervised pre-training were spatially dependent [65].

Such a spatial analysis is appropriate here because we have endeavored to remove all other randomness from the model and are instead analyzing the systems that remain: those of the pre-trained neural model and the mixture of random and functional dynamics that are endemic to the storm system [65]. The mathematical definition of a variogram is as follows [13].

$$\gamma(h) = \frac{1}{2} Var\left[Z(x+h) - Z(x)\right]$$

where γ is called a *variogram*, $Z(x)$ and $Z(x+h)$ are regionalized random variables, which in our case would be the incoming node dot products for the networks corresponding to each location, and x and $x+h$ are the spatial positions separated by h [13].

Figures 5.8 through 5.10 show three examples of variograms that were produced during this procedure. Each of these charts uses a different scale on the ordinate, since the scale of each of the pairwise differences differed significantly. To remove the influence of the scale in the expression of the patterns in the data, we individually scaled each of the variograms to a range between 0 and 1. This allowed us to pairwise compare the patterns that were in the variograms, rather than the data that generated them.

5.5.2.4 Hierarchical Agglomerative Clustering Since all of the variograms were now on the same scale, we used hierarchical agglomerative clustering (HAC) to determine inter-relatedness among variograms. We assumed that similar general shapes for the distributions of the variograms meant that there were similar, or analogous, processes that generated them. Since the number of clusters is an input parameter to HAC, we assessed each clustering from 2 clusters through 9 clusters. To determine what we regarded as an optimal clustering we used the Calinski-Harabasz

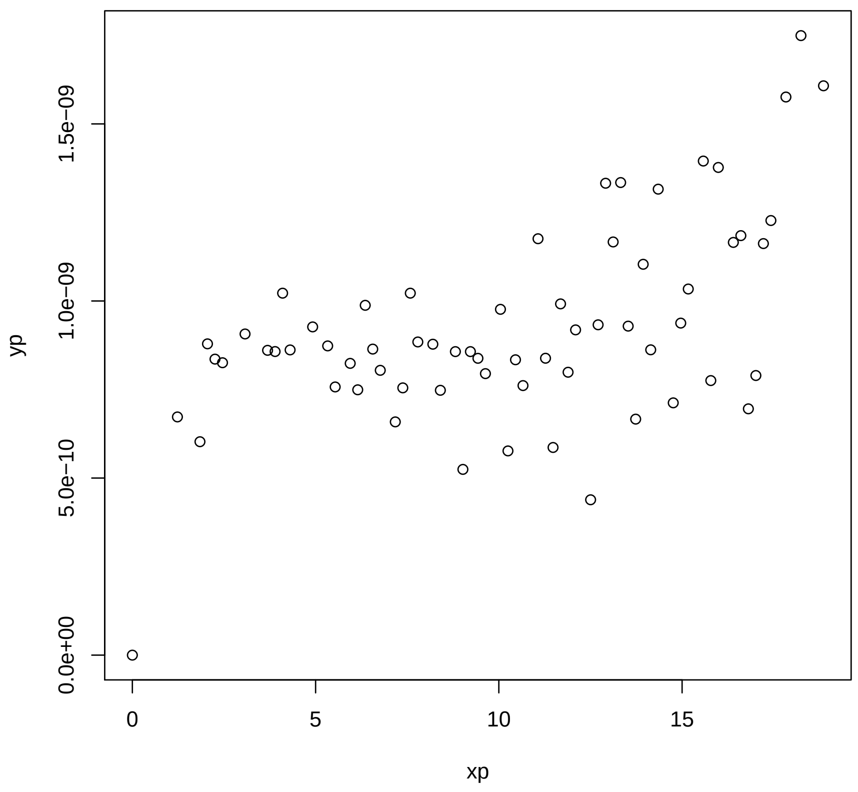


Figure 5.8: Example variograms from each cluster in selected clustering for node 19 (average silhouette coefficient = 0.11).

score (also known as the pseudo-F score) [54] as follows.

$$F_{NC} = \frac{\sum_{i=1}^{NC} n_i d(c_i, c)/(NC-1)}{\sum_{i=1}^{NC} \sum_{x \in C_i} d(x, c_i)/(N-NC)}$$

where NC is the number of clusters, N is the number of objects in the dataset, n_i is the number of examples in the ith cluster, $d(x, y)$ is the Euclidean distance between x and y, c_i is the center of the ith cluster, and c is the center of the dataset. The

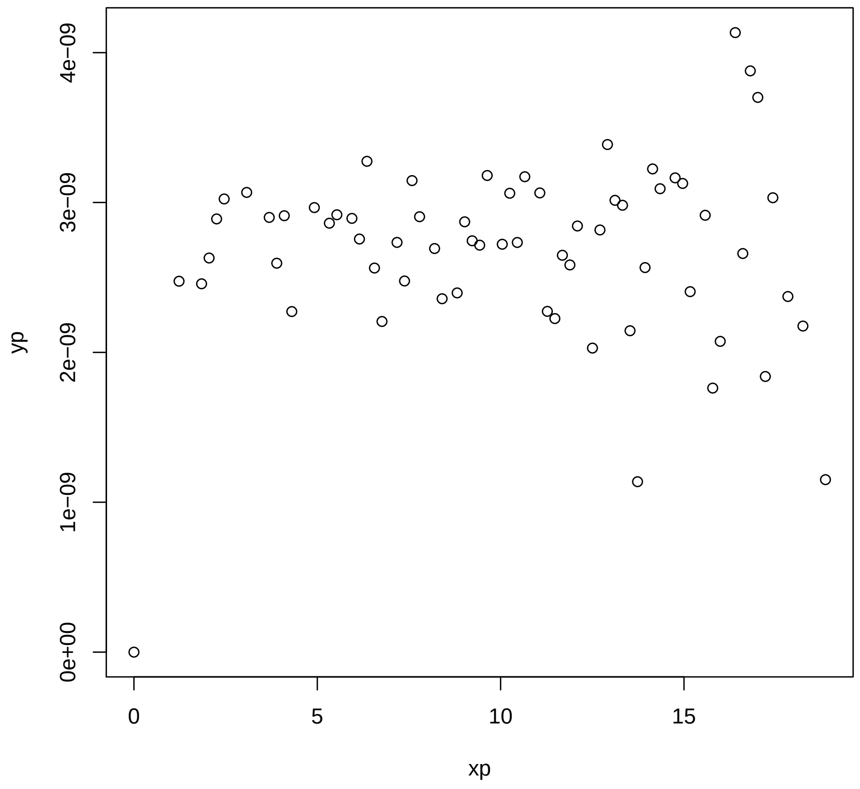


Figure 5.9: Example variograms from each cluster in selected clustering for node 1 (average silhouette coefficient = 0.18).

Calinski-Harabasz measure is an evaluation of cluster validity based on intra-cluster distance and inter-cluster distance. An example Calinski-Harabasz score profile from our training is shown in Figure 5.11. The chart shows that, for this configuration, the optimal HAC clustering to use is three clusters.

We identified and distinguished the clusters inside the clusterings by their silhouette scores, which is a quality measure based on pairwise differences of between

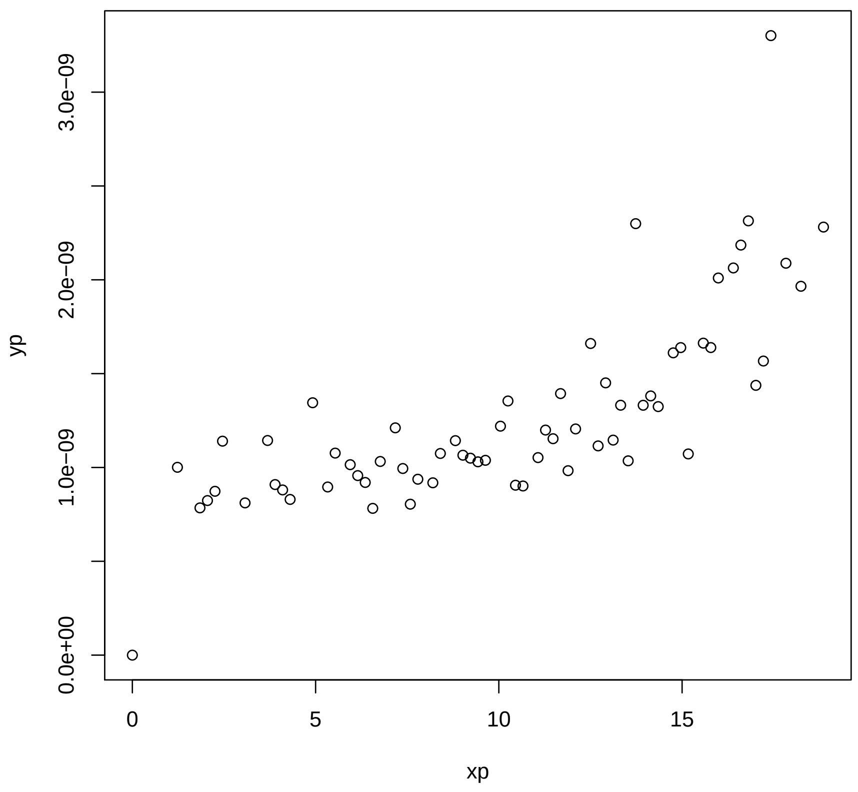


Figure 5.10: Example variograms from each cluster in selected clustering for node 18 (average silhouette coefficient = 0.43).

and within-cluster distances. The silhouette score is defined as:

$$Sil_{NC} = \frac{1}{NC} \sum_{i=1}^{NC} \left[\frac{1}{n_i} \sum_{x \in C_i} \frac{b(x) - a(x)}{\max\{b(x), a(x)\}} \right]$$

where

$$a(x) = \frac{1}{n-1} \sum_{x \in C_i, y \neq x} d(x, y)$$

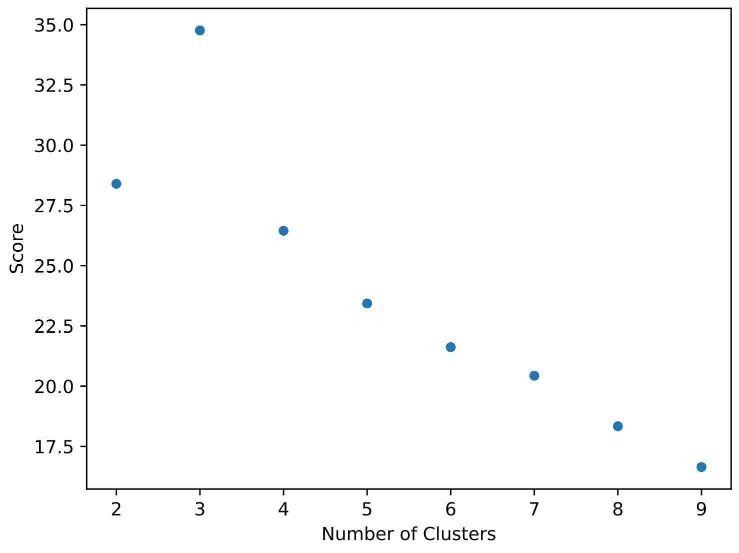


Figure 5.11: Calinski-Harabasz score for clusterings: 2–9 Clusters

and

$$b(x) = \min_{i,j} \left[\frac{1}{n_j} \sum_{y \in C_j} d(x, y) \right].$$

After using the Calinski-Harabasz score to select the number of clusters as an input parameter, the clusters that were produced using HAC had different silhouette scores. We used that as an indication of cluster quality, which varied widely. A maximal silhouette score is to be preferred over minimal, so we wanted to see if cluster quality in this respect had an impact on the resulting convergence and prediction performance. The results depicted in Figures 5.16, 5.17, and 5.18 are divided by average silhouette score for each cluster. Again, NC is the number of clusters, n is the number of objects in the dataset, $d(x, y)$ is the Euclidean distance between x and y, and c_i and c are the centers of an individual cluster and entire dataset respectively.

The three charts in Figures 5.8 through 5.10 are variograms of nodes that were examples of three clusters, each with different average silhouette coefficients (ASC).

Algorithm 5.2 Fixed node identification

```
 1: procedure FNID($locations, ae\_proto, data$)
 2:     $trained\_ae \leftarrow []$
 3:     for all $l \in locations$ do
 4:         $trained\_ae = trained\_ae + train(ae\_proto, data_l)$
 5:     end for
 6:     $ae\_dp \leftarrow []$
 7:     for all $ae_l \in trained\_ae$ do
 8:         $dp_{ae_l} \leftarrow []$
 9:         for all $node\_a \in nodes(ae_l)$ do
10:             for all $node\_b \in nodes(ae_l)$ do
11:                 $dp_{ae_l} = dp_{ae_l} + dot\_product(weights(node\_a), weights(node\_b))$
12:             end for
13:         end for
14:         $ae\_dp = ae\_dp + dp_{ae_l}$
15:     end for
16:     for all $node \in nodes(ae\_proto)$ do
17:         $semvars \leftarrow []$
18:         $semvars = semvars + variogram(ae\_dp[node.id])$
19:     end for
20:     $dendro = dendrogram(semvars)$
21:     $ch\_dict \leftarrow \{\}$
22:     for all $cut\_levelin(2 \ldots 9)$ do
23:         $ch\_dict[cut\_level] = calinski\_harabasz(dendro[cut\_level])$
24:     end for
25:     $max\_cut\_clustering = dendro[max\_ch(ch\_dict)]$
26:     $asc \leftarrow \{\}$
27:     for all $cluster \in max\_cut\_clustering$ do
28:         $asc[cluster] = avg\_silhouette(cluster)$
29:     end for
30:     return $max\_sil(asc)$
31: end procedure
```

When we examined the variograms across the space of all nodes, we observed this variety of patterns. It is this data that allows us to separate nodes that we fix in the next step, as opposed to nodes that we allow to vary.

The fixed-node identification procedure is shown in Algorithm 5.2. The parameters shown on Line 1 are all of the locations in the investigation, the prototype network (*ae_proto*), and all of the data corresponding to all of the locations. On Lines 3 through 5 *ae_proto* is cloned for each of the locations and each clone is trained on the data from that location. Lines 6 through 15 show the collection of pairwise dot products of all the incoming weights for all nodes for all of the trained autoencoders. The variograms are created on Lines 16 through 19. On Line 16 each node in the prototype is extracted from the prototype only for the use of identifying the other nodes in the other networks. Since all of the networks started out as clones of the prototype, this is so that all corresponding nodes can be extracted from each of the other networks using *node.id* on Line 18. On that same line, the function $variogram(ae\_dp[node.id])$ creates a variogram of the corresponding node's pairwise dot products. Line 20 creates a dendrogram, or HAC, of all of these variograms. The Calinski-Harabasz score for each cut level of the dendrogram is obtained in Lines 21 through 24 to determine the optimal clustering. This optimal clustering is extracted on Line 25, where the function *max_ch* extracts the dendrogram from the *dendro* list that has the highest Calinski-Harabasz score. Lines 26 through 29 create a dictionary where the average silhouette score of the dendrograms contained in each cluster are collected. Finally, on Line 30 the cluster of nodes that have the highest silhouette score in the optimal clustering is returned.

Though there are several loops in this algorithm, its time complexity is dominated by the collection of the pairwise dot products in Lines 6 through 15. The time complexity of these lines is $O(l \times n^2 \times p)$, where l is the number of locations, n is the number of nodes, and p is the size of the previous layer.

Considering the hierarchical agglomerative clustering, our technique of determining which is more organized vs. which is less organized is expensive, as it involves creating a dendrogram, at a cost of $O(l^3)$, where l is the number of locations, and cutting this dendrogram at several levels. The aggregate cost of this procedure thus

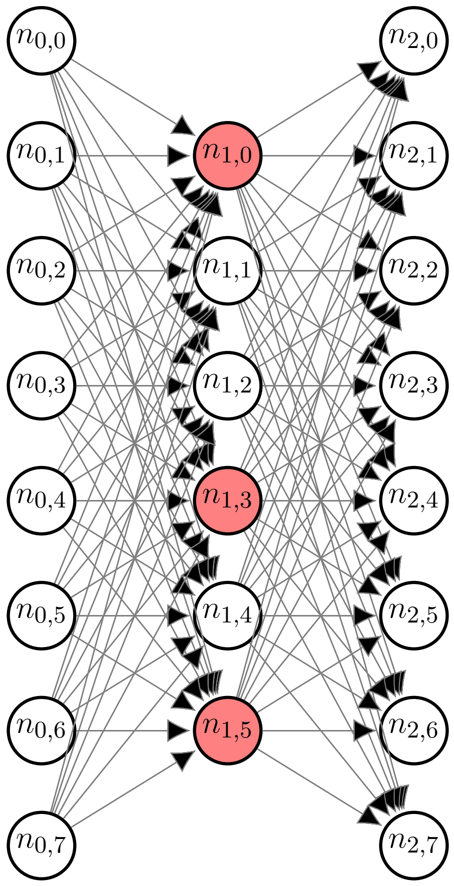


Figure 5.12: Autoencoder fixed nodes

ends up as $O(d \times n \times l^3)$, where d is the number of data instances and n is the number of hidden nodes. But given this is used to support analysis rather than training, this added cost should not be regarded as problematic.

5.5.2.5 Fixed Pre-Training What we obtained from each cluster is the *fixed-set*, which is a list of nodes to transfer to other locations in the dataspace. This forms the basis for the transfer learning experiment. In this procedure, the original autoencoder layers were once again copied from the single prototype autoencoder. The node weights for the fixed-set of nodes were then transferred into the copy of this prototype. The resulting autoencoder layers were pre-trained in this configuration, holding the weights of the fixed-set constant throughout the pre-training procedure (disallowing any change weight during the backpropagation procedure). Figure 5.12 shows this situation for one layer of the network. In this figure, the shaded (red) nodes are copies from another network identified from a cluster of spatially correlated feature detector nodes. In this paper we only used a one-layer stacked autoencoder; however, we intend to extend this to multiple layers in future work.

For the "Surrounding POI Experiments" described in Section 5.6.1, we refer back to Figure 5.5. The location in the upper left indicated by the star shows an example AOI whose node weights were copied, as they were identified as holding transferrable information. The surrounding dots represent the areas of interest to which these node weights were copied and the network's performance tested.

For the "Linear Cross Transfer Experiment," whose results are described in section 5.6.2 we refer to Figure 5.20, where the weights were copied from each corner location (locations 12, 19, 82, and 89) into the networks corresponding to the line of AOI's leading to the respective diagonal's opposite corners. For example, for location 12 in the figure, the fixed-set was copied to the networks to be trained on trained on data from locations 23, 34, 45, 56, 67, 78, and 89.

5.5.3 Fine Tuning and Testing

The stacked autoencoder reassembly and test procedure is depicted in Figure 5.13. The beginning of the procedure has some elements in common with the procedure shown in Figure 5.7, with the prototype networks being replaced by the Fixed Pre-Trained Networks. The remainder of the diagram shows the prediction process is situated.

5.5.3.1 Network Assembly

Figure 5.14 shows the first hidden layer copied from the Autoencoders. The weights are allowed to vary during fine-tuning, allowing

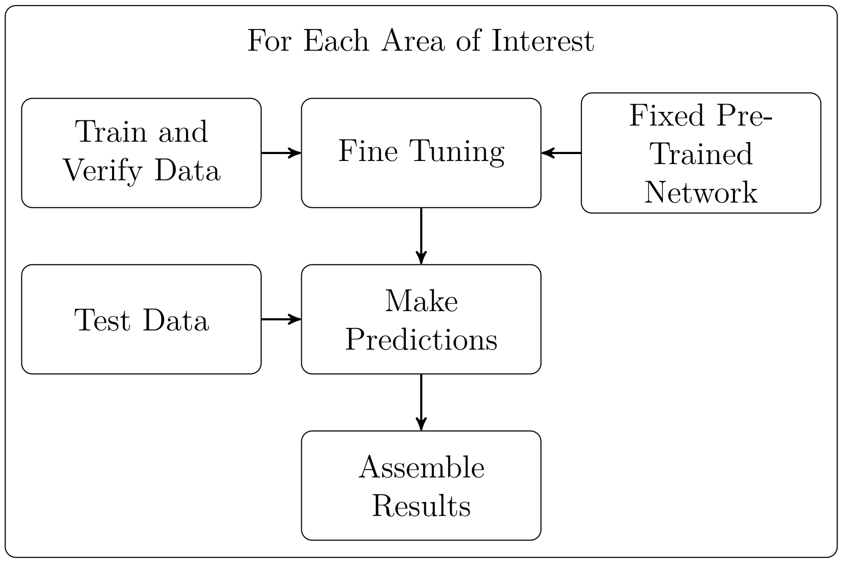


Figure 5.13: Fine tuning and testing

each network to more fully represent its own area of interest. The previously fixed nodes are shown in red to show that this layer was copied from its fixed pre-trained representation.

5.5.3.2 Fine Tuning and Testing After transfer and during fine-tuning, the training data are fed into the network in the same order as they were fed in for the unsupervised pre-training procedure. This, again, is to remove as many sources of variation as we could during the entire procedure. The number of iterations and associated mean squared error of the networks were tracked and compared to the original training to determine if the transfer learning process was more efficient and effective as hypothesized.

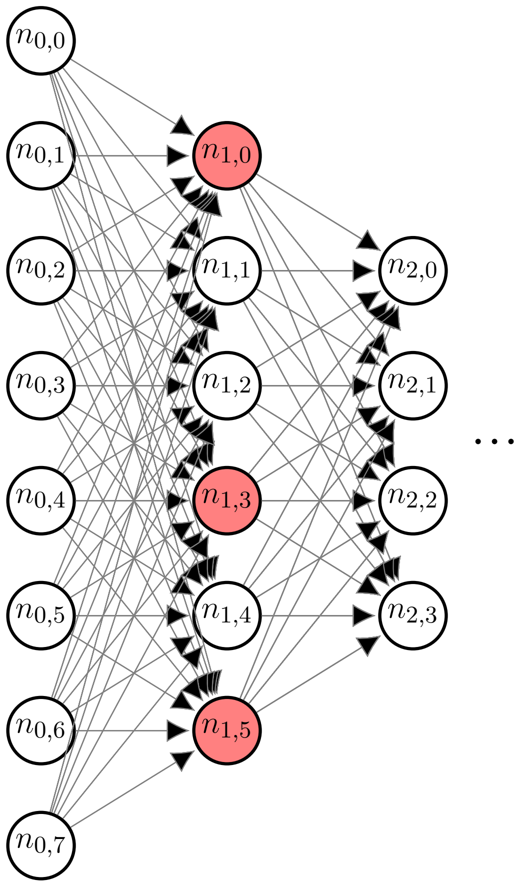


Figure 5.14: Stacked autoencoder fixed nodes

5.6 Results

5.6.1 Surrounding Point of Interest Experiments

Figure 5.15 shows the difference in convergence time that is typical for each of the locations surrounding the location from which we transferred the fixed-set. As can be seen, fixing the nodes from the pre-training of the center cell had the effect of substantially reducing convergence times. It also shows consistently lower overall mean squared error with respect to autoencoder reconstruction during the pre-training procedure.

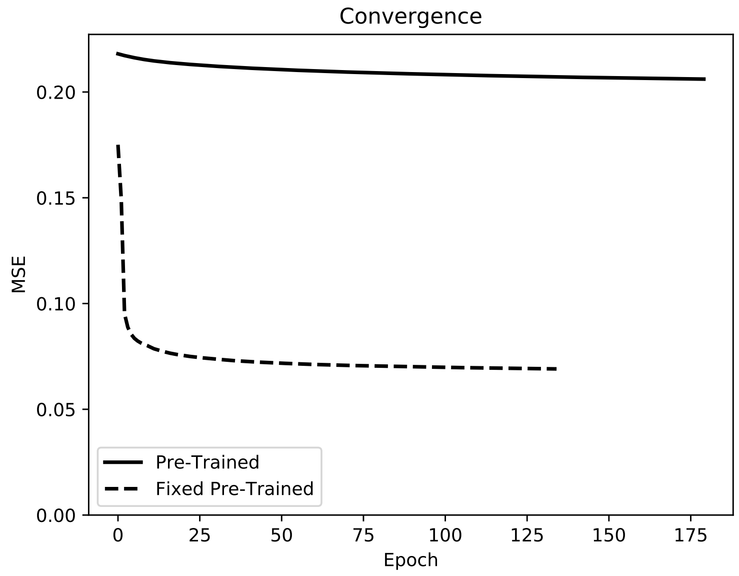
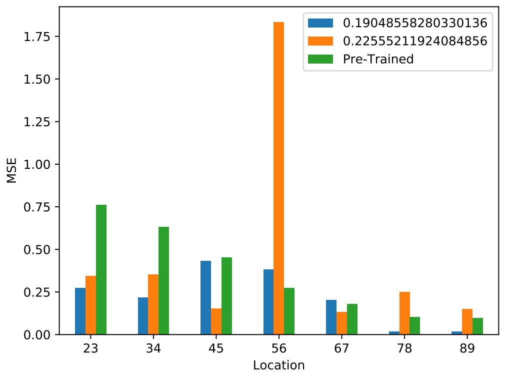


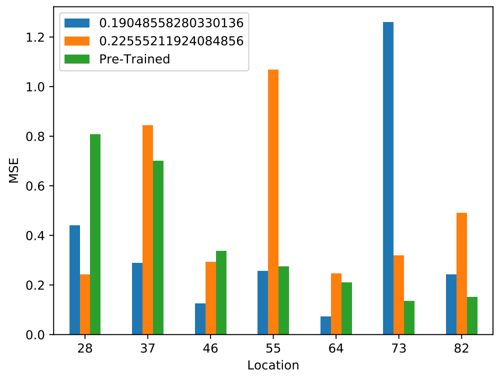
Figure 5.15: Typical convergence comparison plot

Figures 5.16, 5.17, and 5.18 show the test performance when predicting the wind vectors for each of four configurations that we used. We tested configurations that were both regularized (L1 regularization) and unregularized, and we used both the ReLU and hyperbolic tangent (tanh) activation functions in the networks that were assembled from the autoencoder labels. Of particular interest is the configuration that used regularization and the ReLU activation function. In this configuration, using fixed nodes from the cluster corresponding to the higher average silhouette coefficient produced better predictive results, in general, than either those of the fine-tuned pre-trained networks or the fixed pre-trained configuration using the lower average silhouette coefficient. In general, the results from the fixed pre-trained configurations corresponding to the lower silhouette coefficients produced more erratic results.

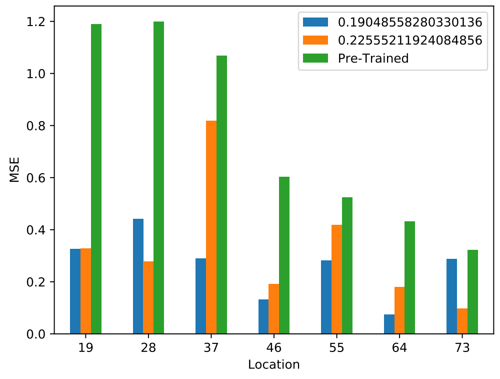
The heatmaps in Figure 5.19 are visual representations of the average difference in predictive accuracy for the 8 cells surrounding the cell from which the fixed-set



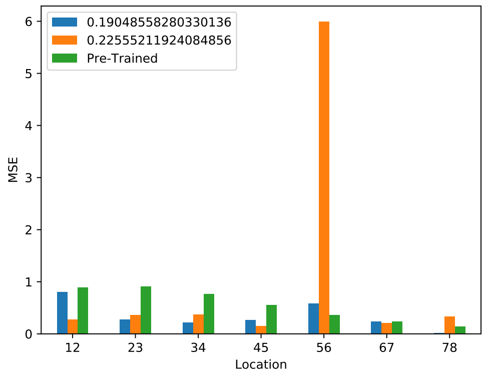
u component, from loc 12



u component, from loc 19



u component, from loc 82

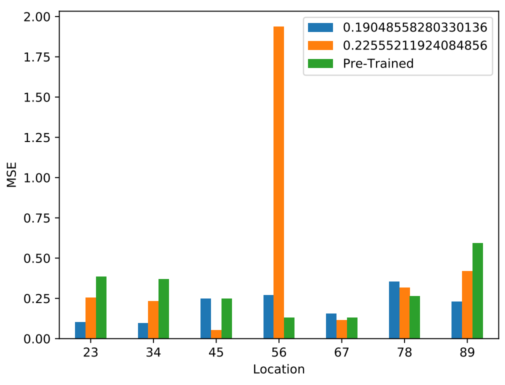


u component, from loc 89

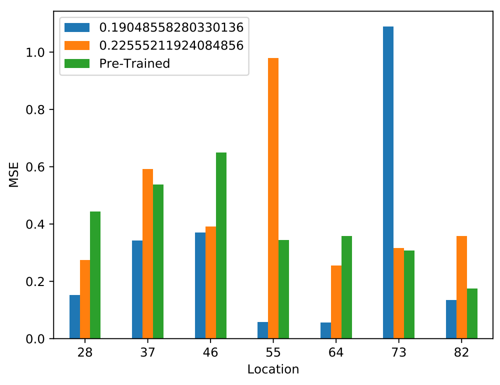
Figure 5.16: u Component cross location prediction performance for ReLU configuration with regularization: silhouette coefficients given in legends

was copied. Again, referencing Figure 5.5, the fixed-set was copied from the location represented by the star. We created the heatmaps by averaging the differences of the MSE's of each location surrounding the starred location.

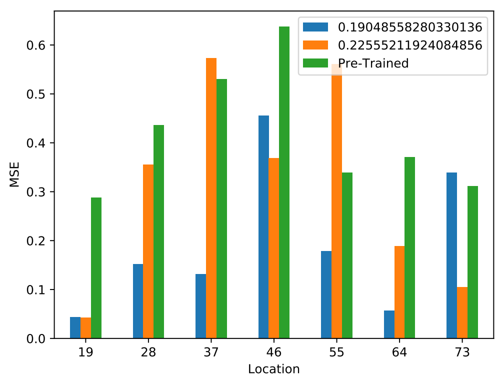
The first row of heatmaps represents the cluster with the lowest silhouette coefficient, and the second row represents the cluster with the highest silhouette coefficient. In these heatmaps, we observe a greater mean squared error variability in the cluster with the lowest silhouette coefficient. This may suggest that greater



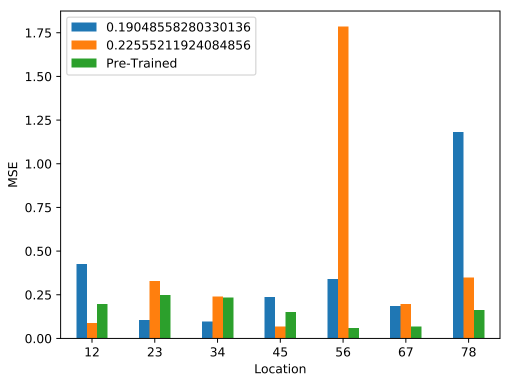
v component, from loc 12



v component, from loc 19



v component, from loc 82



v component, from loc 89

Figure 5.17: v Component cross location prediction performance for ReLU configuration with regularization: silhouette coefficients given in legends

transferability is achieved using the fixed-set from the cluster with the greater silhouette coefficient. For the u component, these differences are shown numerically in Tables 5.3 and 5.4.

5.6.2 Linear Cross Transfer Experiment

In this experiment we wanted to see how the convergence properties and prediction accuracy changed as we moved far across the data space. To achieve this, the fixed-sets were taken from the networks corresponding to the corners and

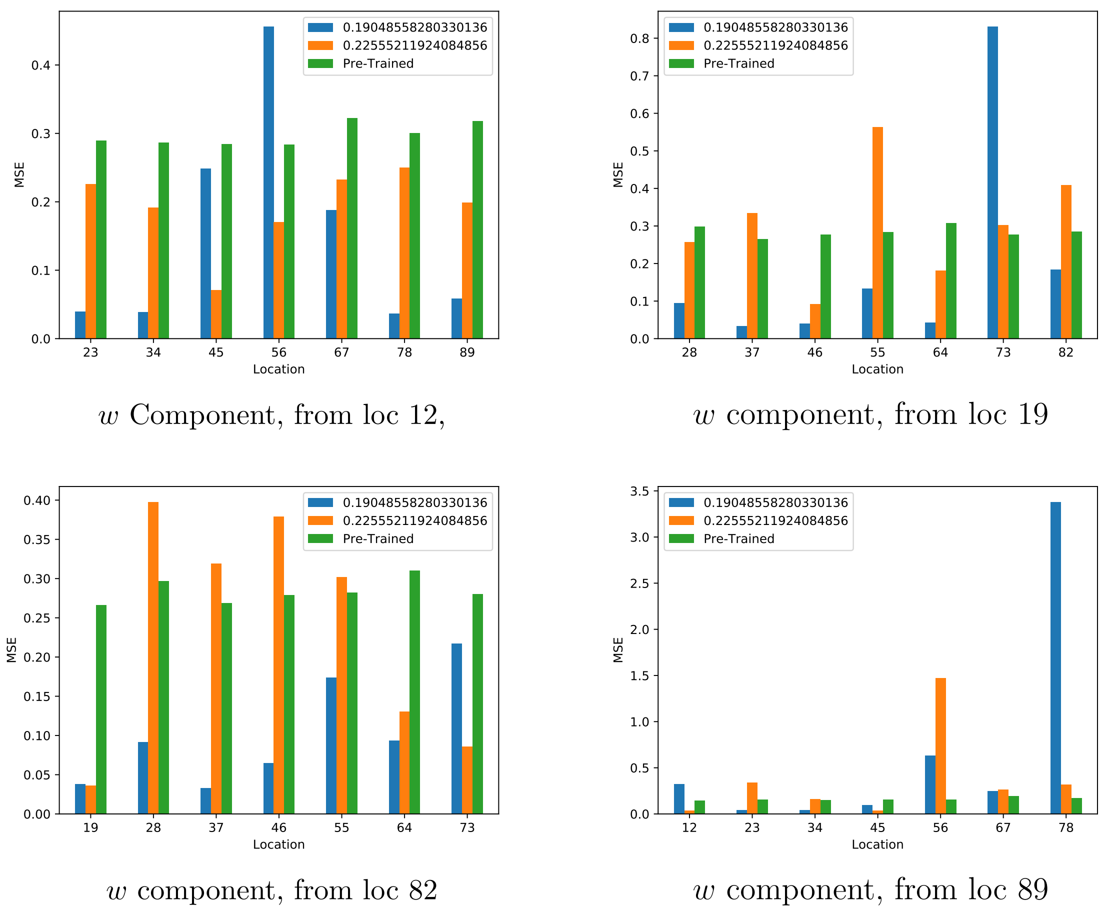


Figure 5.18: w Component cross location prediction performance for ReLU configuration with regularization: silhouette coefficients given in legends

copied to the networks corresponding to the line crossing the data space diagonally, as shown in Figure 5.20. For example, nodes were copied from location 12 to each network along the path to location 89, etc.

The convergence behavior resembled that from the previous experiment, whose results are shown in Figure 5.15, with minor variation. The convergence time was substantially lower and the MSE converged to a similarly lower value.

Table 5.3: Differences in prediction error in surrounding regions for locations in cluster with silhouette coefficient 0.190

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0.071663	0.181648	0.091526	0.144587	0.142470	0.107549	0.144603	0.156638
1	0.173422	0.198090	0.120792	0.098738	0.134753	0.216859	0.180250	0.236516
2	0.146268	0.065116	0.107992	0.053564	0.069530	0.102851	0.129786	0.104743
3	0.152467	0.104508	0.108294	-0.031736	0.024594	0.015864	0.134914	0.167747
4	0.193729	0.172512	0.074459	0.117674	0.123894	0.061272	0.065049	0.122773
5	0.126530	0.127691	0.147620	0.155785	0.126462	0.115997	0.148730	0.251933
6	0.083317	0.183645	0.200183	0.159251	0.192128	0.101274	0.035824	0.133931
7	0.027401	0.126359	0.077443	0.208089	0.099937	0.045052	0.044794	0.148031

Table 5.4: Differences in prediction error in surrounding regions for locations in cluster with silhouette coefficient 0.226

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	-0.006103	0.026441	0.031365	0.079269	0.075486	0.113234	0.089962	0.010246
1	-0.046888	0.002377	-0.035044	-0.038244	-0.001453	0.061009	0.093929	0.087198
2	0.034286	0.017958	0.001052	-0.040874	-0.016071	0.062118	0.022195	0.059957
3	0.097574	-0.001716	-0.068291	-0.085860	-0.106535	0.004428	0.137219	0.085019
4	0.057002	0.008859	0.029313	0.019012	-0.083162	-0.054874	-0.026438	-0.000454
5	-0.009035	-0.054717	-0.061529	-0.013007	-0.050549	-0.055664	0.049241	0.116108
6	0.079068	0.029965	0.058726	0.026723	-0.056399	-0.039050	-0.068245	0.040383
7	-0.002633	0.000531	-0.000337	0.049586	-0.034041	0.017514	0.008220	0.102648

Figures 5.21 through 5.23 shows the prediction performance of the resulting networks for the configuration where no regularization was used and ReLU was used as the activation function. For the first two rows the locations moving to the right across the x axis indicate a movement in the dataspace farther away from the network from which the trained nodes were copied. In the last two rows, movement to the left along the x axis indicates this movement.

5.7 Discussion and Contributions

In this chapter we developed a carefully controlled experiment to analyze the transferability of knowledge stored in a neural network across a complex data space. In our experiment, we performed functional data registration on all the input and output dimensions. Then we initialized a single autoencoder clone for each area of interest. During the training we did not allow random sampling of the data and we restricted the batch size to one. We did this to reduce the differences in scale and to control the training of the networks in such a way as to evaluate what each area of interest where the learned features are aligned across the structures of the networks. Exercising greater control over the autoencoder training procedure allowed us to compare the resulting feature detectors across all locations. This analysis method is unique in the neural network transfer learning domain in that it operates on functionally registered data, creates consistent representations of the feature space using autoencoders, and analyzes the effects that distance has on the features extracted.

One contribution that we made in this chapter is our controlled analysis method. We wanted to have as much control as possible over the training process. This allowed us to compare the components of the neural networks side by side. In this case we were most interested in the results of the unsupervised pre-training, as these results contain the feature detectors of the autoencoder. This means that the encoding layer is subject to the same random initialization, so each autoencoder has the same starting point. Furthermore, the data is sampled in exactly the same way to make sure that the training process is deterministic and consistent across all of the spatial regions in the space.

As another contribution, when analyzing transferability across the data space, we found it advantageous to align time and magnitude in the functions being analyzed. This way the shape of the functions could be compared as if the common cause of the function's behavior happened to each location at the same time and characterized each area in the same way. Referencing our firetruck example from Section 5.4.2, there may be ways that the observers affect the perceived sound of the firetruck as well, such as their position relative to buildings and other obstacles. In other words, the environment and context that defines the data space must be regarded as a critical part of the process that generated the data. This is the case in many situations such as neurological data in the form of the propagation of brain wave patterns, sociological data in the form of community changes in society, and epidemiological data in the form of propagating disease through a population [multiple citations (and explanations) needed].

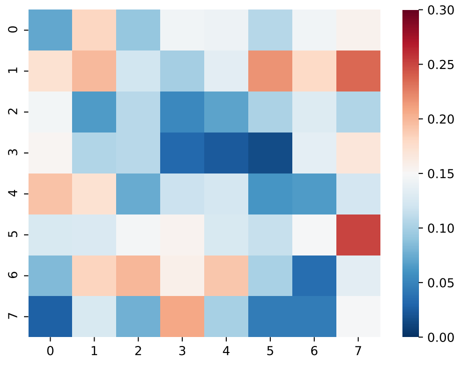
We also contributed an analysis method uses variograms from the field of spatial statistics in a new way, focusing on the their shapes as an indicator of whether or not a trained autoencoder node captures useful information. Simply, we assert that the more difficult it is to explain the shape of the variograms corresponding to a trained node, the less well-trained that node is. This is the motivation behind the "more organized" vs. "less organized" description given in Section 5.5.2.4 and behind clustering these variograms based on their shapes.

5.8 Conclusions and Future Work

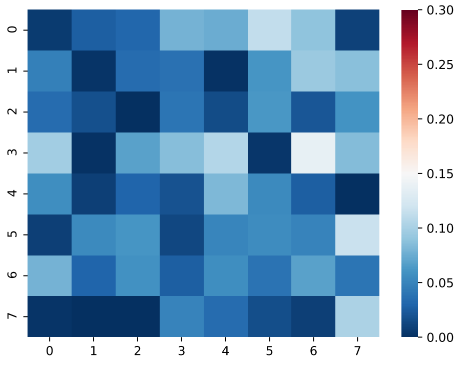
We observe that selecting a clustering from a HAC clustering with the highest Calinski-Harabasz score, and using this clustering with the information about the silhouette scores of the clusterings can provide information about what random initializations allowed better generalization across the dataspace, given the context of the other initializations. The procedure of copying the weights corresponding to the nodes in the clustering to the original randomly-initialized network, and keeping those weights constant during pre-training results in lower convergence times for pre-training. We note, however, that this procedure is not practical in the sense of providing a general training and transfer process. This is by design, as coming up with such a procedure was not our goal. Rather, our goal was to analyze spatial correlation in learned feature detectors to determine whether or not such spatial correlation might exist and be exploited in transfer learning.

For future work, we will shift our focus from analysis to developing a practical training procedure. Specifically, our intent is to apply model reduction strategies (e.g., hidden node pruning) to determine those feature detectors in the network that are likely to exhibit the desired spatial correlation. An alternative approach might be to apply the ideas from the lottery ticket hypothesis [31, 93], as a way of identifying transferable subnetworks. This hypothesis states that there are subnetworks within large networks that, when trained in isolation, "can converge in a comparable number of iterations to a comparable accuracy." It focuses on locating these subnetworks, which have "won the lottery" by being initialized in an advantage part of the initialization space. These features might then be transferable and tunable in the other areas of the dataspace.

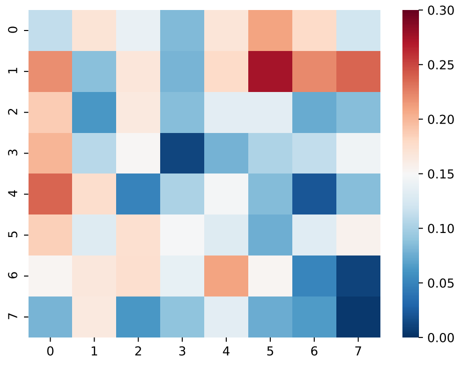
An interesting component to this problem is in determining whether or not spatial differences in this type of data cause the behavior of the attendant phenomena to be different enough to warrant an original network for each specific area to be modeled. We do not think this is so, but if we are right then there must be a way to divide the dataspace intelligently into areas appropriate for application of models that were trained on "compatible" areas. For this, we also need to determine what we mean when we call these areas “compatible.” There also may be a need to allow overlap, in which case we may introduce fuzzy clustering or a type of mixture model.



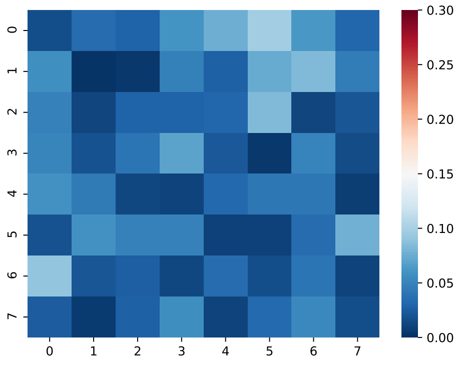
u component, SC = 0.190
Avg RMSE: 0.1240 ± 0.0029



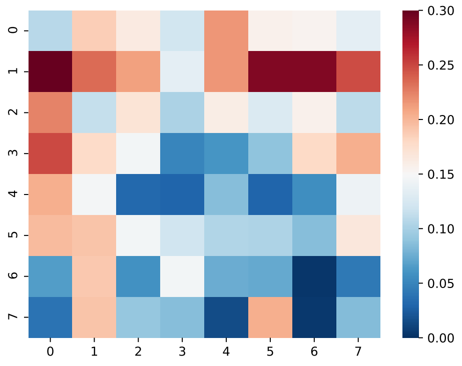
u component, SC = 0.226
Avg RMSE: 0.0456 ± 0.0012



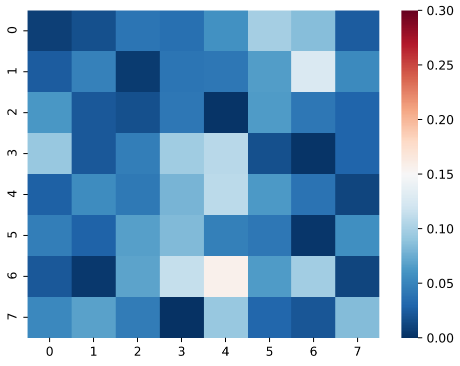
v component, SC = 0.190
Avg RMSE: 0.1292 ± 0.0035



v component, SC = 0.226
Avg RMSE: 0.0365 ± 0.0006



w component, SC = 0.190
Avg RMSE: 0.1364 ± 0.0053



w component, SC = 0.226
Avg RMSE: 0.0507 ± 0.0012

Figure 5.19: Heat maps depicting neighborhood mean squared error differences: Regularization, ReLU

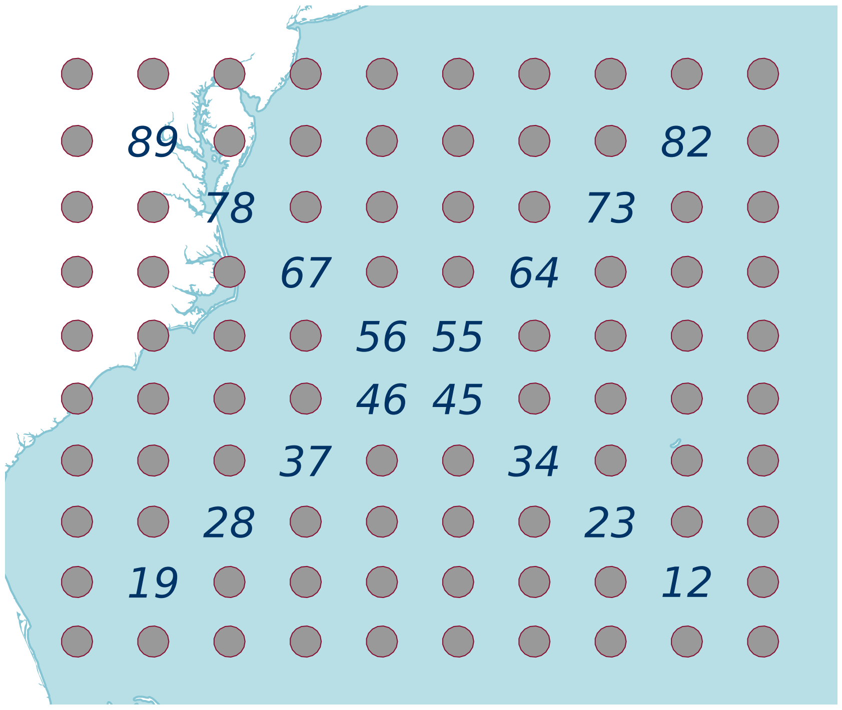
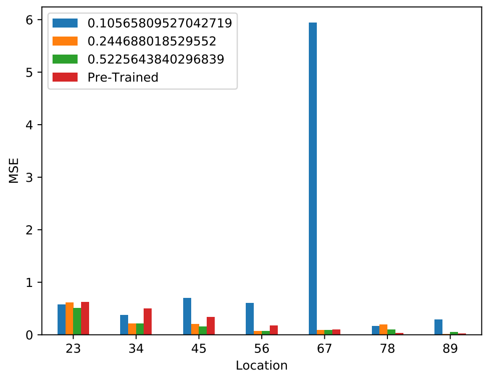
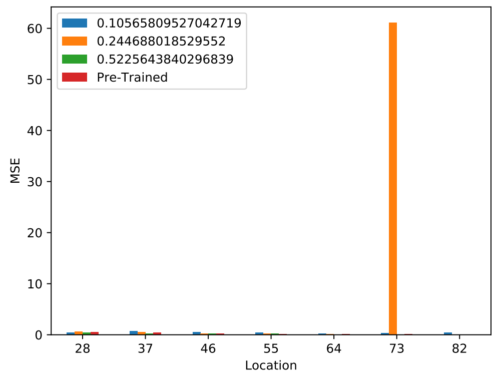


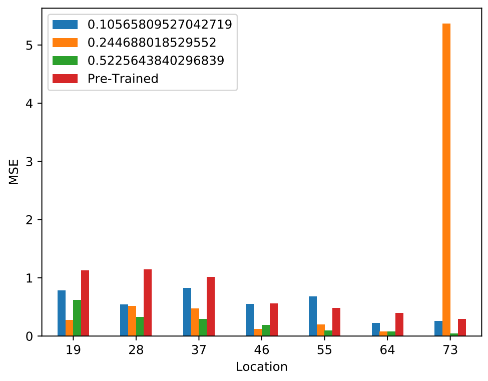
Figure 5.20: Linear cross training locations



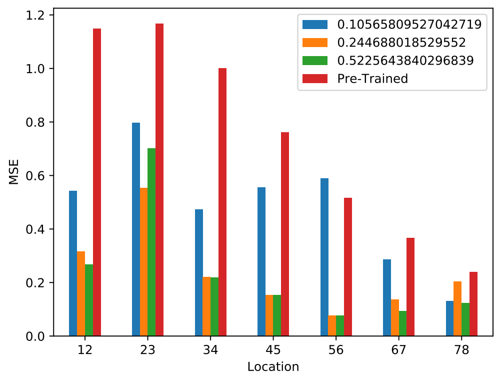
u Component, From Loc 12



u Component, From Loc 19

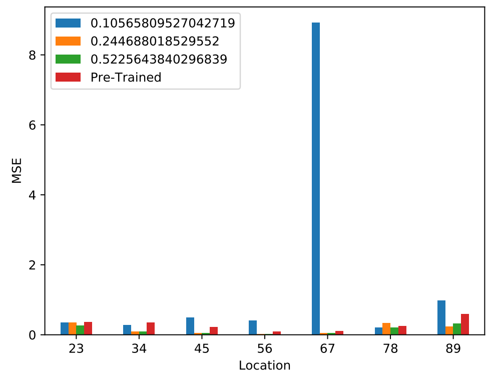


u Component, From Loc 82

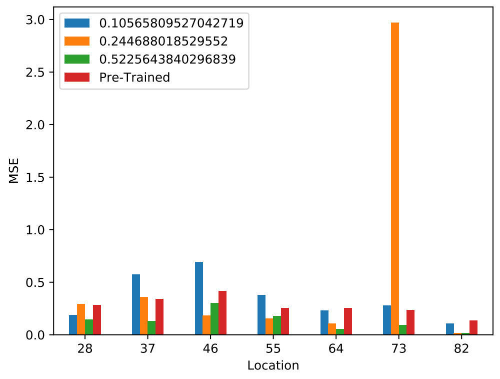


u Component, From Loc 89

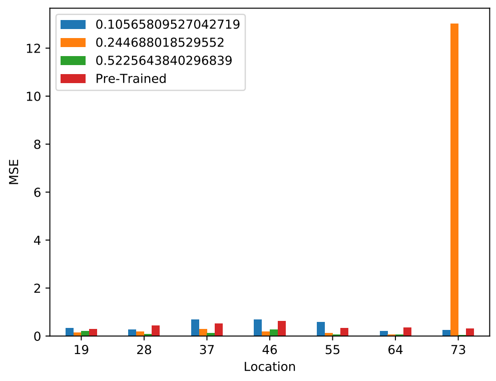
Figure 5.21: u Component cross location prediction performance for ReLU configuration without regularization: silhouette coefficients given in legends



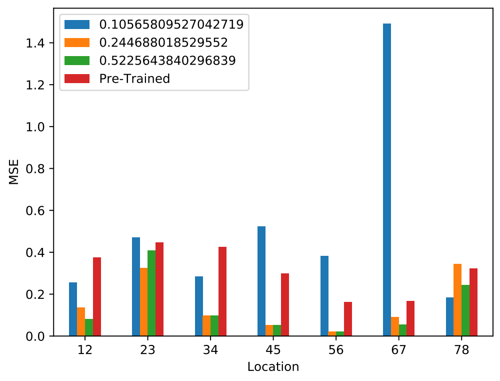
v Component, From Loc 12



v Component, From Loc 19

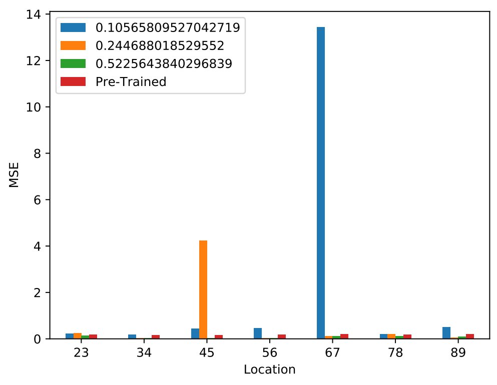


v Component, From Loc 82

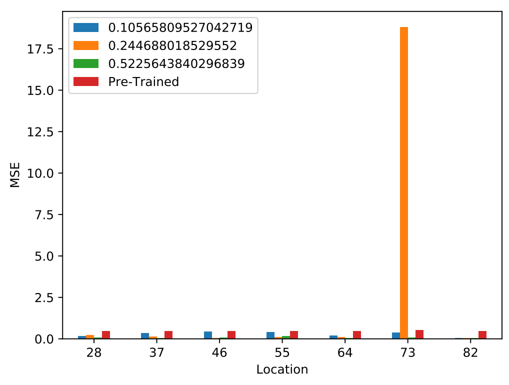


v Component, From Loc 89

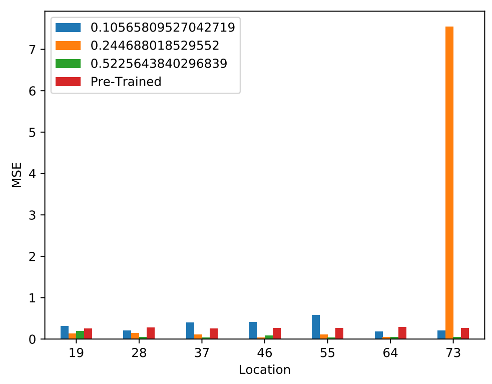
Figure 5.22: v Component cross location prediction performance for ReLU configuration without regularization: silhouette coefficients given in legends



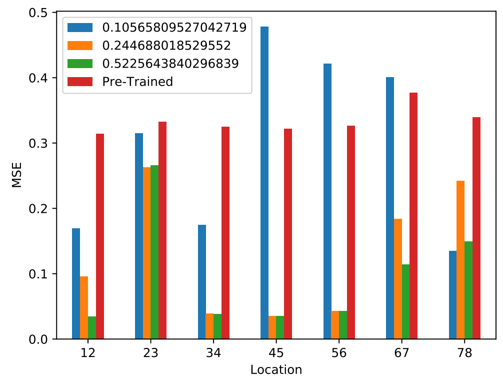
w Component, From Loc 12



w Component, From Loc 19



w Component, From Loc 82



w Component, From Loc 89

Figure 5.23: w Component cross location prediction performance for ReLU configuration without regularization: silhouette coefficients given in legends

CHAPTER SIX

SPATIAL FEATURE EXTRACTION

6.1 Introduction

As an intermediate step in understanding knowledge learned by ANN's it is important to be able to make use of information gained in training for novel but related purposes. For us, this means having the ability to transfer subsets of internal nodes that have been trained to make effective predictions on a related set of data. This concentrates our effort on preserving accumulated intelligence rather than interpreting the model.

A variety of problems exist where one might consider training a single model to perform predictions of the associated dataspace. Situations exist where a single model to cover the space is suboptimal; it would be better to train several models covering subsets of the space, albeit not as an ensemble where predictions are combined. These situations often arise when the data is spatial or temporal in nature. The challenges then arise as to determining the relevant subsets of the data for a particular model to cover, efficiently training the set of models, and then determining which model to apply at runtime.

In this chapter, we proceed from the perspective of working specifically with spatial data[1] This makes the third challenge trivial, thus our focus is on addressing the first two. To that end, we explore how to preserve information encoded in ANN's trained in specific spatial regions and then identify key hidden nodes in networks

[1]Technically, the data is spatiotemporal; however, we train over the entire temporal range, thus abstracting out the temporal relationships.

to be used to train networks in spatially similar regions. The main question being addressed is whether it is possible to preserve knowledge encoded in the hidden layers of several ANN's and collect that knowledge in a network that can effectively make predictions over a more granular subdivision of the entire dataspace.

The remainder of this chapter is organized as follows: In the next section we discuss some background information relevant to our task, including some related work and a characterization of the data that were used here. Section 6.3 details the approach we take as well as summarizing the common concepts that we utilize in our approach. Our experiments are described after this in Section 6.4 with the results presented in Section 6.5. Finally, in the last section, we present our conclusions based on the results and outline directions for future work.

6.2 Background

6.2.1 Related Work

In this work we use autoencoders to pre-condition our hidden layers via greedy, layerwise unsupervised pre-training [26,66]. Autoencoders perform feature extraction by exploiting a property of being "undercomplete," meaning that they ideally capture the most salient features of the training data at a lower dimensionality [39]. Networks pre-trained in this manner have been credited with causing the renewed interest in deep neural networks starting in 2006 [17,39,43,76].

The data that we are interested in is spatiotemporal functional data, which is data that exist in a continuous space whose changes happen as a function of space and time. The field of functional data analysis is characterized and explored by Silverman and Ramsay in [89]. A critical characteristic of this data is its tendency to vary in a continuous, non-abrupt manner.

Layerwise relevance propagation (LRP) is a procedure for removing some of the “black box” characteristics of artificial neural networks [12,62,84,85]. Originally, LRP was intended to attribute contributions of single pixels in an image to classification decisions that a neural network made. It has also been used to compute scores for regions of images “denoting the impact of the particular image region on the prediction of a classifier” [18].

Partitioning of the dataspace was addressed by Rakhmatova, et al. in [75]. The authors cite differences in relief, soil type, terrain, and anthropogenic effects causing the data to be “spatially inhomogenous” as a motivation for their work. They use three strategies for partitioning the data: random partitioning into 70% training and 30% test partitions, an approach based on spatial quotas (minimum number of observations per partition), and an approach based on spatial quotas taking the modeled variables’ magnitude difference into account. This is consistent with our motivations in this dissertation. However, though they address pre-partitioning a dataset for greater accuracy in the application of ANN’s, their method mainly addresses optimally splitting the data for training and test purposes, rather than to create a multiplicity of networks to more accurately make predictions on their own subdivisions, as is done here.

In [88], Sergeev, et al. refer to processes that cause “spatial heterogeneity” in studying the spatial distribution of topsoil components. They used spatial components as inputs to their procedure, allowing the ANN’s that were trained on the data to learn how the spatial locations affected the function being simulated. Our approach does not rely on the ANN’s to do this and instead uses separate networks for the contiguous related regions identified via the two stages of clustering detailed below.

6.2.2 Data

For this study, we used the Microwave CubeSat fleet simulation dataset created by Zhang and Gasiewski [109]. This data depicts radiometric and wind readings from one day over Hurricane Sandy, an Atlantic Ocean hurricane in 2012. The average spatial resolution of the data is 5 kilometers, which we down-sampled during the spatial binning process to 15 kilometers. The temporal resolution was 15 minutes. The configuration of the data for training and testing purposes was the same as was used in Chapter 5.

The particular task that we use to study our procedure is the same as in Chapter 5, which is to infer the u component of the storm's wind vector (East-West component) using radiometric data. The data that we use is a hybrid of two datasets (radiometric and wind) that were taken over the same area at the same time. In Chapter 5, we explore the functional relationship between radiometric data and wind vector data. This is useful to the meteorological community since radiometry does not measure wind directly.

Table 4.2 shows the input features that were used and output for these experiments. These are the same features used in Chapter 5; however, we considered all three components in that work (i.e., u, v, and w) but we restrict our attention here to the u component alone. This decision was made in the interest of time. As can be seen, the input dimensions do not directly address the wind vector but do address components that are somewhat related to it.

Over a large enough space on the Earth, the further apart two distinct areas are the more the data from these areas will be influenced by different forces. To address this question we use weather data, which is a form of spatial-temporal-functional data. Over a large enough space on the Earth, the further apart two distinct areas are the more the data from these areas will be influenced by different forces. This will cause the data to behave in fundamentally different ways, motivating the use of several ANN's to make predictions over this dataspace. It is in this area that we make our investigation.

6.3 Approach

6.3.1 Overview

The general approach we take is to divide the dataspace into contiguous clusters based on two types of representations. The primary clustering is based on the original training data (data instance clustering). The intent of data instance clustering is to provide a first-look agglomeration of contiguous areas with similar data behavior over the training instances. The second set of clusters is based on the LRP values of the hidden nodes of a stacked autoencoder-based multilayer perceptron that has been trained on the respective clusters that make up the primary clustering (LRP clustering). The intent of LRP clustering is to capture similar knowledge that has been learned by the networks and provide a basis for hidden-node generalization.

Our approach to knowledge transfer among networks is to identify nodes in the hidden layer that have been most significant in producing correct predictions. These nodes are then combined with other nodes from other networks within their spatial cluster to form a hybrid network. Since we are performing regression, we are interested in studying the effect of both excitatory and inhibitory neurons.

6.3.2 Computational Components

6.3.2.1 Spatial Data Clustering

To exploit functional relationships in the data, we developed a Locality-Sensitive Hierarchical Agglomerative Clustering algorithm (LSHAC). This proceeds from the assumption that points of interest that are close to one another (spatially) behave functionally with a greater degree of similarity

Algorithm 6.1 Locally-sensitive hierarchical agglomerative clustering

```
 1: procedure LSHAC($L, cluster\_limit$)
 2:     for all $l \in L$ do
 3:         $c_l \leftarrow l$                                   ▷ Each location is its own cluster.
 4:     end for
 5:     while $num(C) > cluster\_limit$ do
 6:         $l_r \leftarrow rand(L)$
 7:         $c_{l_r} \leftarrow$ cluster containing $l_r$
 8:         $closest\_locs = []$
 9:         for all $l_{c_{l_r}} \in c_{l_r}$ do
10:             $L_a \leftarrow adjacent(l_{c_{l_r}}) \backslash c_{l_r}$
11:             $closest\_locs \leftarrow closest\_locs + \operatorname*{argmin}_{l_a \in L_a}(distance(l_a, l_{c_{l_r}}))$
12:         end for
13:         $c_{l_r} \leftarrow c_{l_r} + closest(\_locs)$
14:     end while
15: end procedure
```

than points that are further apart. As mentioned in Section 6.1 these areas may be influenced by common factors that are relatively alien to other areas. Assuming this, we begin by performing LSHAC on the raw training data points.

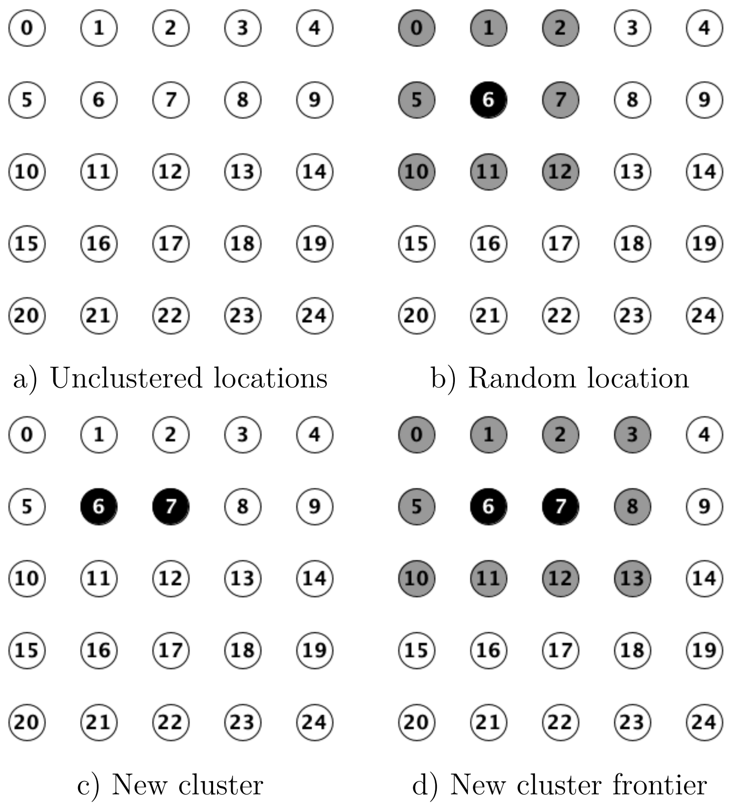
The key difference between LSHAC and normal hierarchical agglomerative clustering is that, at each agglomeration stage, we only consider the locations neighboring a randomly selected neighboring cluster. Note that we refer to these locations as Areas of Interest (AOI). We use cosine similarity to compare these neighboring locations to determine which of the neighbors is added to the cluster. Pseudo-code for our procedure is shown in Algorithm 6.1. This is done in stages, and the candidates for expansion are selected randomly at each stage.

Algorithm 6.1 shows our procedure for LSHAC. Lines 2 through 4 show that at the beginning each location is its own cluster. The major loop (lines through constrains the number of clusters being formed to the *cluster_limit* parameter. In lines and we select a random location and get the cluster corresponding to that location. Lines through iterate through all locations inside this cluster, saving the closest location outside this cluster to each of these interior locations. The cluster being examined (c_{l_r}) is then merged with the external cluster containing the location that is closest to any location inside c_{l_r} on line .

Normally, Hierarchical Agglomerative Clustering algorithms have a time complexity of $O(n^3)$, where n is the number of locations. For LSHAC, we constrain the similarity search space to include only adjacent clusters, and only adjacent locations within those adjacent clusters. Consequently, this algorithm has a time complexity that is closer to the Adjacency-constrained HAC algorithm described by Ambroise, et al. [10]. Considering only merging clusters that are adjacent constrains the time complexity to that of $O(n \log n)$, which is the number of iterations to merge the clusters into one cluster.

Figure 6.1 depicts the initial stage of clustering using 25 points in a five by five grid. At this stage, no clustering decisions have been made, and each location is treated as its own cluster. For illustration purposes, we select location 6 arbitrarily as a starting point (Figure 6.1b). The surrounding locations of location 6, shown in grey, are identified, and each of their cosine similarities with location 6 is calculated. Now suppose that location 7 is determined to be most similar. Figure 6.1c shows location 7 added to the cluster that includes location 6. Figure 6.1d then shows the next step by considering the surrounding locations of the cluster containing locations 6 and 7.

For the second stage of clustering, Figure 6.2 shows that all of the locations have either been paired with another location or are orphaned by not being included in another cluster (e.g., locations 8 and 18). The second round of clustering begins with collecting all of the locations neighboring a randomly selected cluster. We continue the previous example by using the cluster containing locations 6 and 7, with the



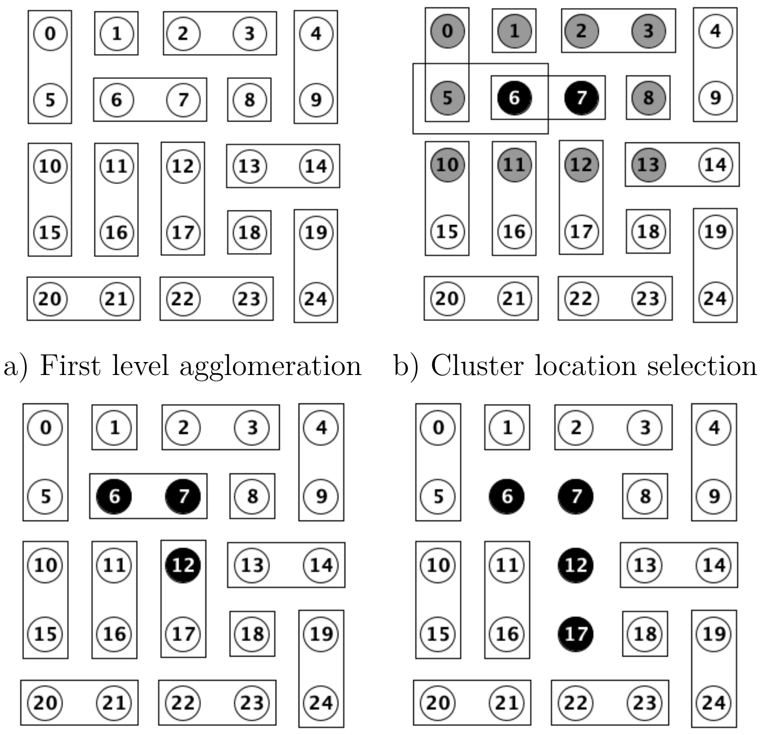
a) Unclustered locations b) Random location

c) New cluster d) New cluster frontier

Figure 6.1: Clustering a 5 × 5 grid of locations

surrounding location shown in grey. Figure 6.2c shows that location 12 has been selected as the location with the highest cosine similarity. Analogous to a single-linkage approach to clustering, the entire cluster containing location 12 is now merged, creating a new cluster containing locations 6, 7, 12, and 17. This new cluster is excluded from evaluation until the second stage completes.

6.3.2.2 Unsupervised Pre-Training and Fine Tuning Each ANN trained here begins by unsupervised pre-training (UPT) of a single autoencoder [32]. The described here should be generalizable to multilayer perceptrons constructed in the same way. We use UPT to find favorable initialization points from which to apply supervised learning with the ground truth data [39].



a) First level agglomeration b) Cluster location selection

c) Location selection d) Cluster merge

Figure 6.2: Clustering a 5 × 5 grid of locations (cont'd)

As was done in Chapter 5, a single initialization across all locations was used to preserve comparability. During UPT, all of the training data for a given cluster was used to train an autoencoder, presented in the same temporal order across all networks. After training, the decoder portion of the autoencoder was discarded.

6.3.2.3 Layerwise Relevance Propagation Because we are interested in the contributions of hidden nodes to train a final regression model, we use LRP at the hidden layers to determine the most relevant nodes. LRP employs a layer-wise conservation principle that assumes the preservation of a propagated quantity between adjacent layers of a multilayer perceptron [85]. The conservation principle requires

$$\sum_i R_i^{(l)} = \sum_j R_j^{(l+1)}$$

where $R_i^{(l)}$ is the relevance associated to the ith neuron of layer l. To obtain a matrix of relevance scores, this principle is applied on a backward pass through the network. The relevance signal is directed proportionally to the subset of inputs that resulted in the corresponding output for each training instance. The relevance score is calculated according to

$$R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j} + \epsilon \text{ sign}\left(\sum_{i'} z_{i'j}\right)} R_j^{(l+1)}$$

where ϵ is a parameter that encourages numerical stability, removing the possibility that the denominator tends to zero and

$$z_{ij} = a_i^{(l)} w_{ij}^{(l,l+1)}$$

where a_i is the activation value of node i and $w_{ij}^{(l,l+1)}$ is the connection weight between nodes i and j between layers l and $l+1$. This causes the lower-layer neurons (those closest to the input) that mostly contribute to the activation of a higher layer neuron (closest to the output) to receive a larger share of the associated relevance.

LRP was designed specifically to analyze input data. We are interested in capturing the relevance of the hidden layer activations, so we modified the LRP algorithm to do this (Algorithm 6.2). On line 2, we create a new network by removing the output layer and using the activations of hidden layer l as the output. We then capture this output A as new input data for the second part of the procedure. On Line 7, we create a new network by removing the input layer from the original network n_{AOI} and using A as the input. We perform the LRP procedure on each of the records

Algorithm 6.2 Hidden layer LRP

```
1: **function** HLLRP($n_{AOI}, D_{AOI,training}, l$)
2:     $n_{front,l} = n_{AOI}$ with layer $\alpha_l$ as output
3:     $A \leftarrow []$                                        ▷ $A$ gets a new list.
4:     **for** $d \in D_{AOI,training}$ **do**
5:         $A[d] \leftarrow n_{front,l}(d)$
6:     **end for**
7:     $n_{back,l} = n_{AOI}$ with layer $\alpha_l$ as input
8:     $\Lambda \leftarrow []$                                        ▷ $\Lambda$ gets a new list.
9:     **for** $\alpha \in A$ **do**
10:        $\Lambda[\alpha] \leftarrow LRP(\alpha)$
11:    **end for**
12:    **return** $\Lambda$
13: **end function**
```

from the A list to obtain the relevance values Λ for each of the internal nodes in layer l. Notic that we run the complete set of training data through the network once, and then the activations through the network another time.

6.3.2.4 Subset Transfer We are interested in capturing the most important subsets of the nodes that make up the hidden layers of our networks. Using these subsets, we create a foundation for pre-training new autoencoders. We demonstrate that this yields multi-layer perceptrons that are more accurate predictors for the local area upon whose data the network has been trained.

When we transfer information among networks, we are combining networks that have already been trained to make predictions within their own spatial clusters. Often, corresponding nodes in different clusters' networks have LRP values that are different but still significant, which is demonstrated in Figure 6.3. Both clusters have produced networks where node $n_{1,0}$ has a significant LRP value.

When we examined the distribution of LRP values, we observed three interesting differences (Figures 6.4 and 6.5). The whisker plot of the LRP values of the nodes from

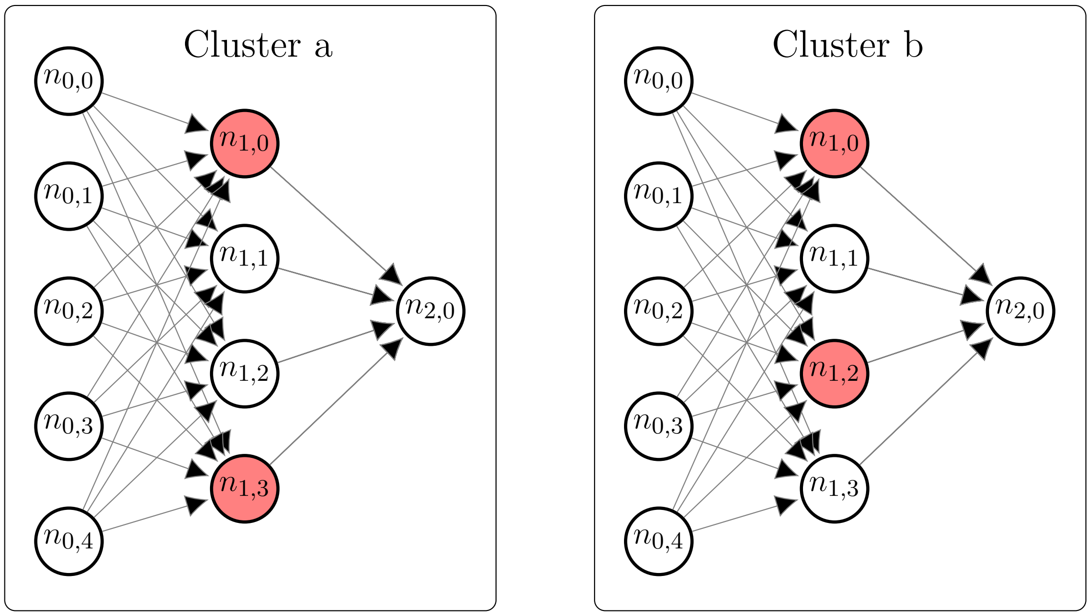


Figure 6.3: Intersecting importances

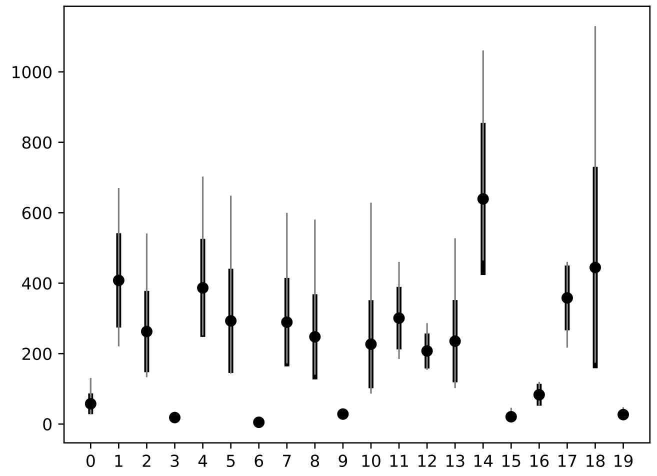
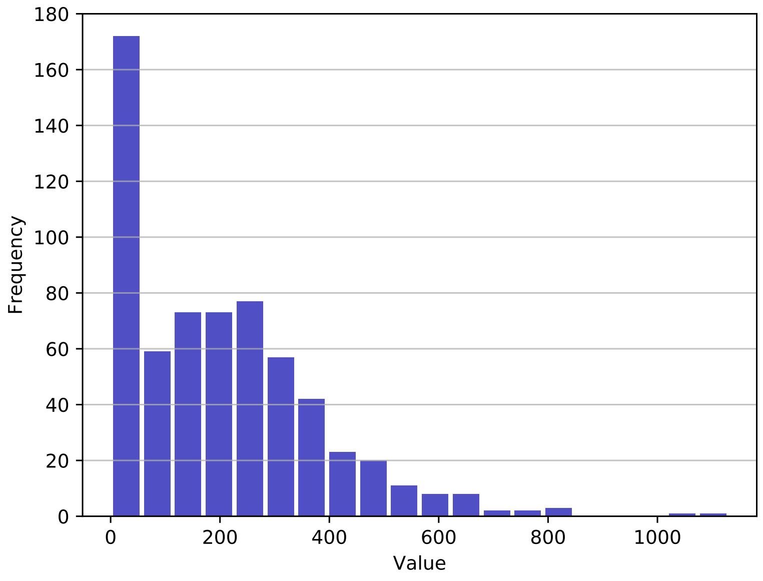
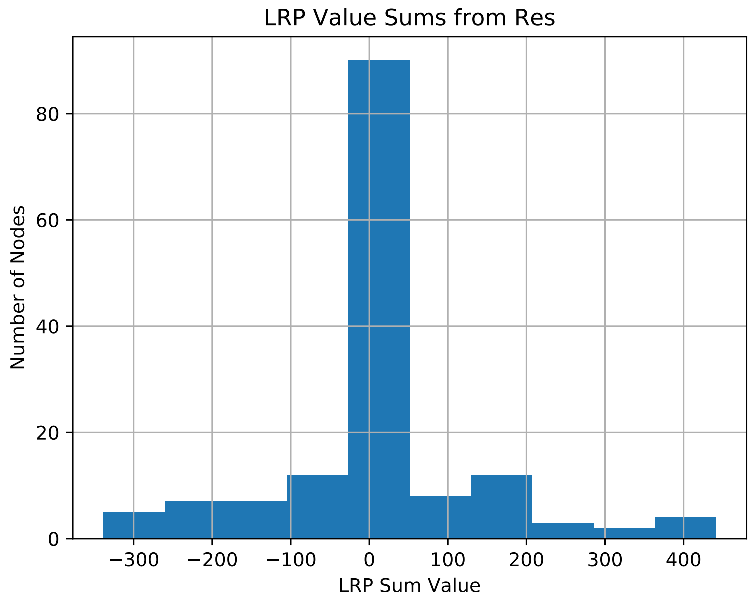


Figure 6.4: LRP value differences

the networks within one cluster (Figure 6.4 shows that there is a variety of differences among the nodes, including many with no difference at all (e.g., nodes 0, 3, 6, 9, 15,



a) Histogram of absolute values

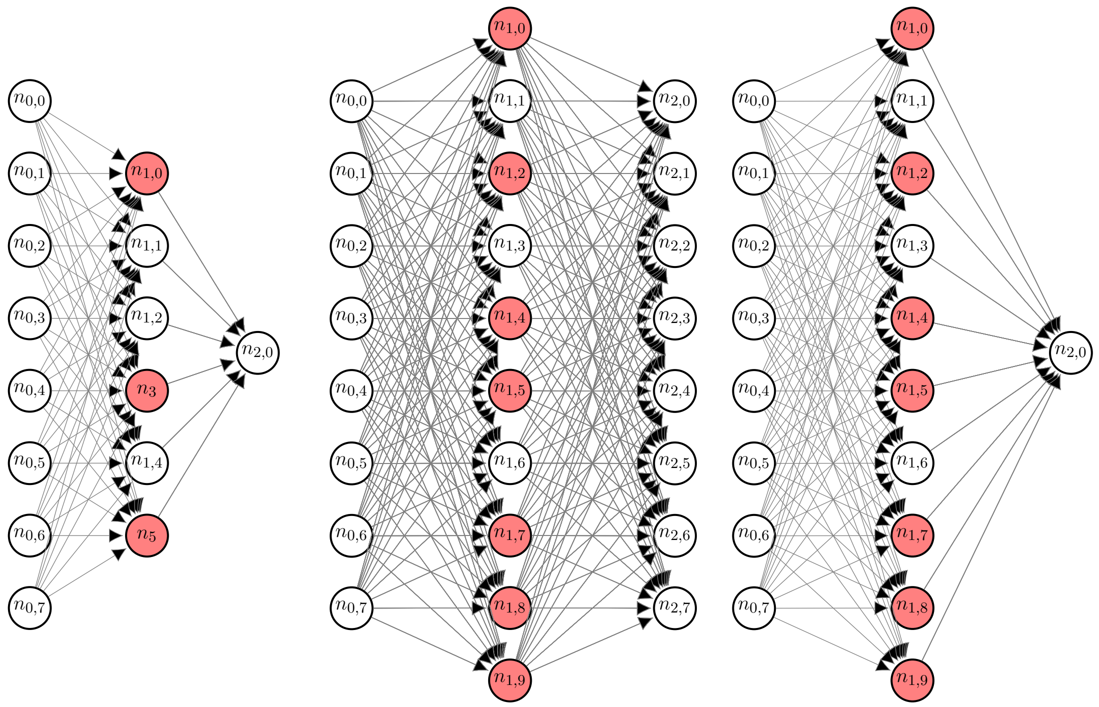


b) Histogram of values

Figure 6.5: Examination of LRP value distribution

and 19). Figure 6.5a shows a histogram of the absolute values of the differences where a large number of nodes have little or no difference with the distribution then trailing off to the right. Figure 6.5b shows a histogram of both the positive and negative LRP values. It should be noted here that the creators of the LRP procedure identified nodes with negative LRP values as having an inhibitory influence on the final prediction. To identify nodes with useful LRP values, we employ "Kneedle", a knee detection procedure to find the point where the influences between the tendency for the LRP sum value to be zero (no influence) and the tendency for the node to have some relevance [86]. The "Kneedle" procedure is based on the idea of finding the points in a curve that are the local maxima for a curve that has been rotated by ϕ degrees about the point (x_min, y_min) through a line formed by the points (x_min, y_min) and (x_max, y_max). We applied this procedure to the final histogram for both increasing (negative) and decreasing (positive) values. Nodes laying outside these knee points were regarded as transferable.

When comparing two networks, if the LRP values of a node in each network are regarded as significant, but one node is inhibitory while the other is excitatory, we split the node in the combined network. Figure 6.6 illustrates this process. In our case we are interested in excitatory and inhibitory nodes because these types of nodes have the effect of either influencing an increase (excitatory) or decrease (inhibitory) the final value of the regression. In this example we only depict the situation where all three of the relevant nodes have been deemed relevant across multiple networks in the merging cluster. Figure 6.6a highlights the three relevant nodes. Figure 6.6b shows that all three have been included in a new autoencoder. The remaining nodes are copied from the untrained template network.


$n_{0,0}$ $n_{0,1}$ $n_{0,2}$ $n_{0,3}$ $n_{0,4}$ $n_{0,5}$ $n_{0,6}$ $n_{0,7}$
$n_{1,0}$ $n_{1,1}$ $n_{1,2}$ n_3 $n_{1,4}$ n_5
$n_{2,0}$
$n_{0,0}$ $n_{0,1}$ $n_{0,2}$ $n_{0,3}$ $n_{0,4}$ $n_{0,5}$ $n_{0,6}$ $n_{0,7}$
$n_{1,0}$ $n_{1,1}$ $n_{1,2}$ $n_{1,3}$ $n_{1,4}$ $n_{1,5}$ $n_{1,6}$ $n_{1,7}$ $n_{1,8}$ $n_{1,9}$
$n_{2,0}$ $n_{2,1}$ $n_{2,2}$ $n_{2,3}$ $n_{2,4}$ $n_{2,5}$ $n_{2,6}$ $n_{2,7}$
$n_{0,0}$ $n_{0,1}$ $n_{0,2}$ $n_{0,3}$ $n_{0,4}$ $n_{0,5}$ $n_{0,6}$ $n_{0,7}$
$n_{1,0}$ $n_{1,1}$ $n_{1,2}$ $n_{1,3}$ $n_{1,4}$ $n_{1,5}$ $n_{1,6}$ $n_{1,7}$ $n_{1,8}$ $n_{1,9}$
$n_{2,0}$

a) Identification of nodes b) Node expansion c) New perceptron

Figure 6.6: Node identification and expansion

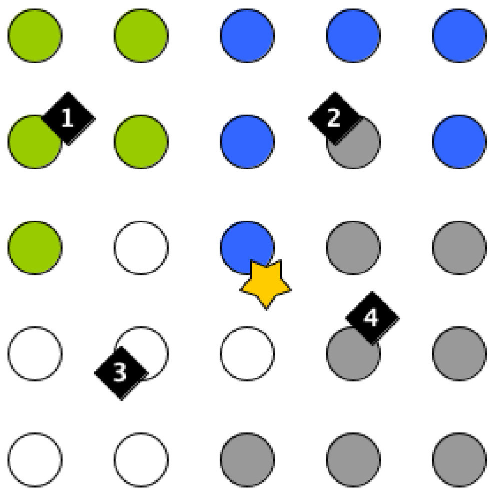


Figure 6.7: Spatially clustered areas of interest with query location

6.3.2.5 Selective Network Application Figure 6.7 shows a 5×5 grid that has been divided into four clusters with the centroids of those clusters depicted as the black diamonds. The centroids are determined by taking the mean of all of the positions in the cluster. Suppose we have a location about which we wish to make predictions (e.g., the yellow star in Figure 6.7). Since centroid 4 is the closest, the network for cluster 4 will be used to make predictions for that location.

6.3.3 Training Procedure

To begin the overall training procedure, spatial data clustering (cf. Section 6.3.2.1) is applied to the raw data from each AOI. We call this "data instance clustering." The targeted number of clusters is an input parameter that guides the algorithm, telling it when to stop clustering. The algorithm stops clustering when the number of clusters is at this limit or when the next agglomeration will cause the number of clusters to be below this limit.

The autoencoders that produce the initial hidden layers for each AOI's network are each copied from a pre-initialized template network. This is done to ensure comparability among each of the networks. This is also important for the aggregation and transfer steps later in the procedure.

Following the unsupervised pre-training, multilayer perceptrons (MLP's) are created using the pre-trained layer as the hidden layer. These MLP's are then fine-tuned with the training data. We then apply the LRP procedure to the trained networks to determine the relevant hidden nodes. The fine-tuned MLP's allow the LRP procedure to occur, since they have been fine-tuned using the ground truth from the training data. Following LRP, a second phase of clustering is performed using the relevance values. It is this clustering step that produces the final clusters among which accumulated knowledge is shared. It is also these clusters that are evaluated using the Selective Network Application procedure from Section 6.3.2.5, which is the final step in the experiment.

The analysis for the transfer procedure occurs after the LRP values have been obtained. We used this information in four ways, each of which is detailed in the extreme lower right cell of Table 6.1. The nodes only above the positive knee point represent the relevant positive nodes. The nodes only below the negative knee point represent the relevant negative nodes.

6.4 Experiments

6.4.1 Experimental Design

For training, testing, and verification, we employed a 10-fold cross-validation strategy where the same time slices are used for each of the splits across all of the networks. The 10-fold cross validation used 90% of the entire set of time slices as the training and validation set The remaining 10% of the entire set of data was used for testing. We further subdivided the training set, with 90% of it used for training and 10% of it for validation. The validation split was used for early stopping to prevent overfitting. The networks were trained to predict wind vector components from the radiometric data collected in the Microwave CubeSat data set.

6.4.2 Experiment Scenarios

Table 6.1 shows the different scenarios of experiments that were run. The global experiment served as a baseline for performance comparison and consisted of training a single network over all of the AOI's. The dual clustering experiments involved combining both data instance clustering and relevance clustering. The data were clustered using the normalized data instances. The LRP procedure was then run to obtain the relevance values for each instance in the training data. For data instance

Table 6.1: Summary of experiment scenarios

Experiment	**Description**	**Variations**
Global	Train one ANN on all data from all locations to predict over the entire dataspace.	N/A
Dual Clustering (DC)	Cluster the dataspace using LSHAC based on the training data. Further cluster the data-space using HLHAC based on LRP values.	N/A
Dual Clustering with Transfer (DCT)	Same as DC. Combine networks using relevance similarity.	1) Use only above positive knee point. 2) Use only below negative knee point. 3) Use both. 4) Use only relevant nodes.

Table 6.2: Experiment configurations for different scenarios

Experiment	**Sub-Experiment**	**Expansion**	**Abbreviation**
Global	N/A	N/A	Global
Dual Clustering	N/A	N/A	DC
Dual Clustering with Transfer	Nodes Above Knee Point	Single	DCT Above Single
”	”	All	DCT Above All
”	Nodes Below Knee Point	Single	DCT Below Single
”	”	All	DCT Below All
”	Above and Below	Single	DCT Both Single
”	”	All	DCT Both All
”	Only Relevant	Single	DCT Rel Single
”	”	All	DCT Rel All

clustering the target cluster numbers were 20, 30, 40, 50, 60, and 70. Following this, the LRP clustering used target cluster numbers of 4, 5, 6, 7, 8, 9, and 10. All 36 pairs were examined. Note that no knowledge transfer among networks occurred in the first set of dual clustering experiments, but the knowledge transfer procedure was applied in the second set.

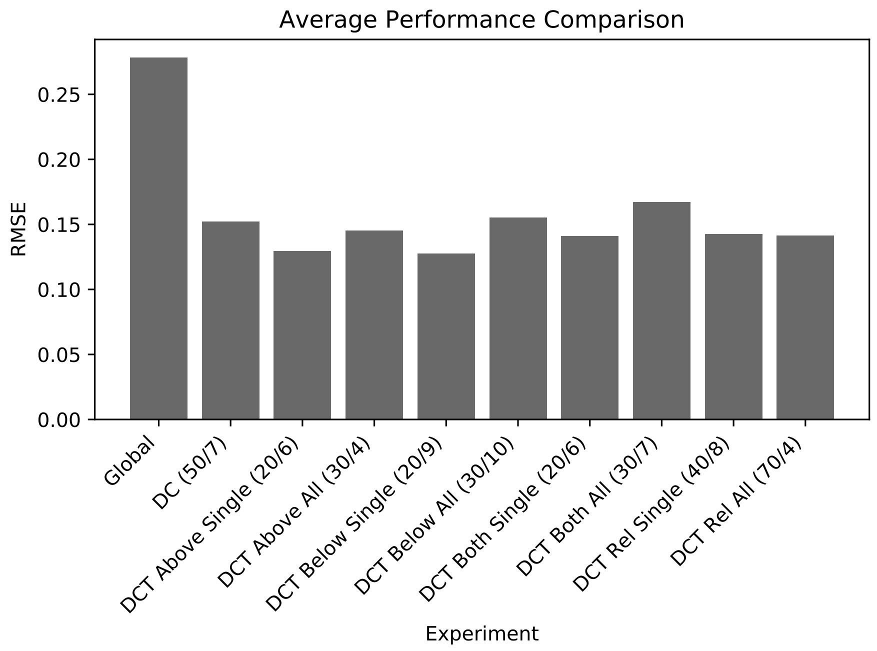


Figure 6.8: Summary of performance

When combining the clusters, there were situations where the candidate clusters for the new combination contained weights that were the same or almost the same. In response to this, the Dual Clustering with Transfer experiments in Section 6.5.3 were subdivided further into two node expansion techniques. In the first case, we simply selected one of the nodes at random, treating it as a representative of the set of nodes with similar weight vectors. Thus, in this case, only one node from the group is transferred to the new network. In the second case, we simply transferred all of the nodes from the collection (resulting in the transfer of as many nodes as were in the group). Table 6.2 shows the abbreviations that are used in the discussion of the experiment scenarios.

6.5 Results

Figure 6.8 summarizes the performance of all experiments in terms of root mean squared error (RMSE) on the wind vector determination task from Chapter 3. As shown, the best performing configurations from each sub experiments all outperformed the Global configuration. The best single configuration was the DCT Both Single (40/4), which had a data instance clustering limit of 40 and an LRP clustering limit of 4.

To test the performance of these scenarios, we ran paired *t*-tests comparing each scenario depicted in Figure 6.8 with the Global configuration. The tests were run on the cross validation error from each of the folds. At an $\alpha = 0.1$ level, we found a significant difference between the Global configuration and all of the experimental configurations, with the exception of the DCT Both All configuration. We also found that the DCT Above Single and the DCT Both Single configurations were significantly better than the Global configuration at the $\alpha = 0.05$ level, with the DCT Both Single configuration having greater improvement over Global. However, when comparing both of these configurations to each other, we found no statistically significant difference, even at the $\alpha = 0.1$ level.

6.5.1 Global Experiment

Results for the Global experiment are summarized in the heat map in Figure 6.9. The global network was initialized with the same weights as for all of the rest of the experiments. Since this experiment was used as a baseline, we only note the general characteristics of the performance for this network. This figure shows better performance in the North of the dataspace and especially in areas that are over land.

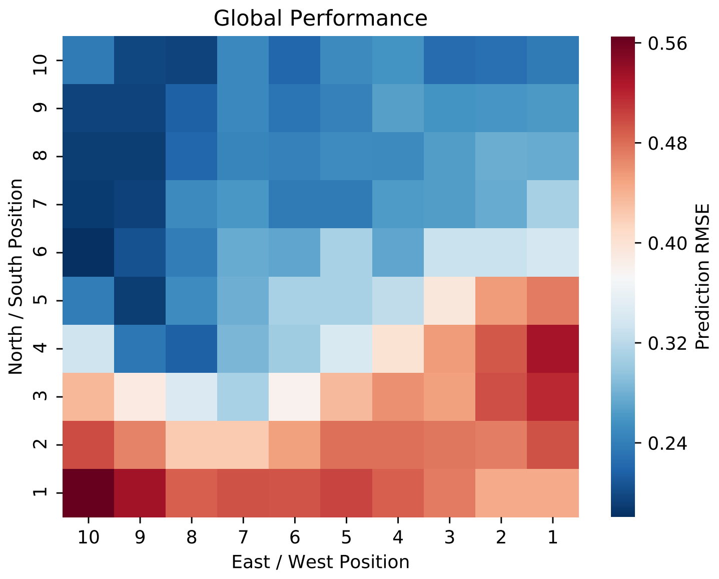


Figure 6.9: Global network performance

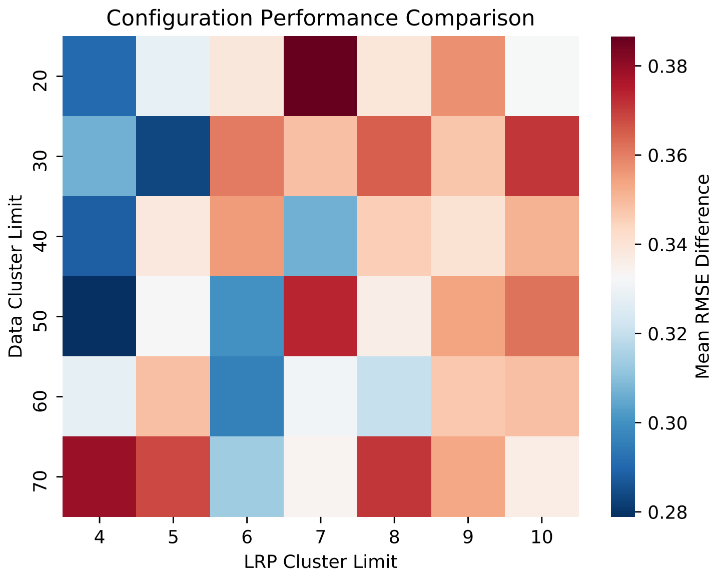


Figure 6.10: Comparing DC scenario comparison to Global

6.5.2 Dual Clustering (DC) Experiment

The differences between the RMSE for the DC experiment and the Global experiment are summarized in the heat map in Figure 6.10. The values used to create the heat map are the mean difference in RMSE over the ten folds for the DC experiment:

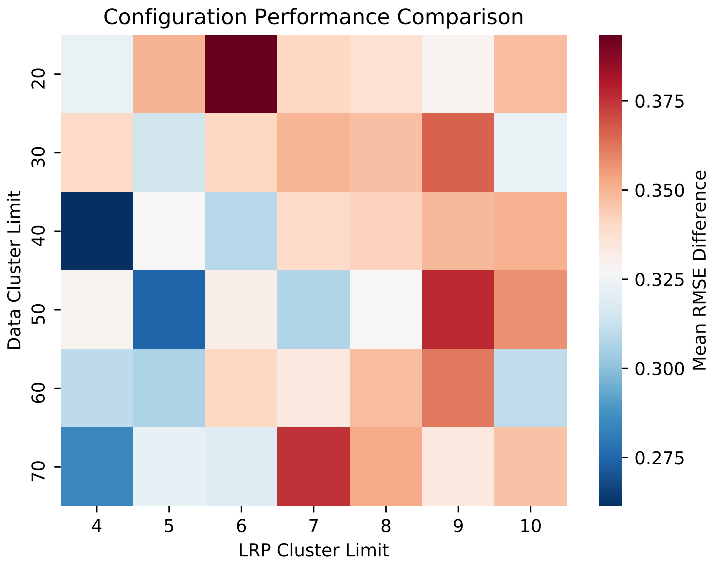
$$\text{HM-Val} = \text{Global-RMSE} - \frac{1}{|folds|} \sum_{f \in folds} RMSE_f$$

This means that the heat map tends towards red when the results favor the experiment and tends towards blue when results favor the Global configuration, with white being neutral.

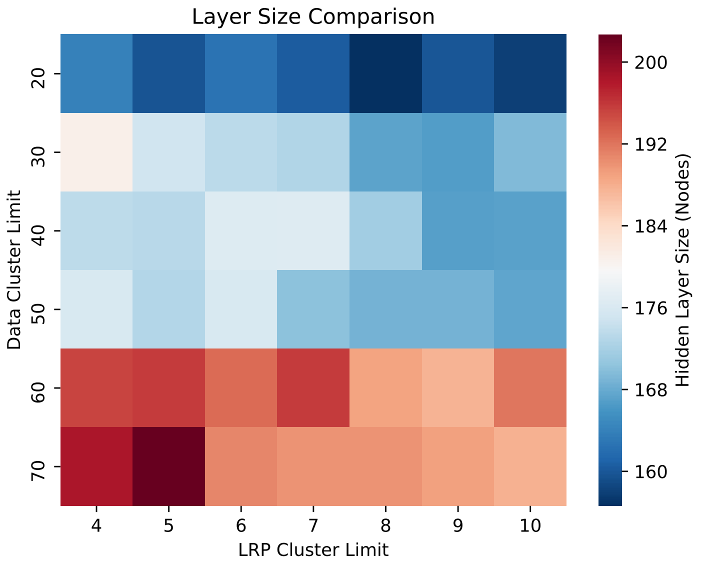
6.5.3 Dual Clustering with Transfer (DCT) Experiment

Figure 6.8 shows the performance of the "DCT Both Single" scenario, which was the best-performing scenario. This heat map is constructed to compare the performance for the 36 different combinations of cluster sizes when configuring data instance clustering and LRP clustering. As shown, the configuration where the cluster limit for data instance clustering was 40 and the cluster limit for the LRP clustering was 4 is the highest-performing version. The heat map in Figure 6.11a is constructed in the same way as in Figure 6.10 and compares the winning "DCT Both Single" configuration to the performance of the Global network. The cell corresponding to the best performing configuration in this diagram (Data Cluster Limit: 40, LRP Cluster Limit: 4) conforms to the lower extreme of the RMSE differences, where the Global average RMSE was 0.330 and the average RMSE for DCT Both Single was 0.261.

Figure 6.11b depicts the the final hidden layer (autoencoder) sized that resulted from the node expansion procedure detailed in Section 6.3.2.4. The winning configuration (again, Data Cluster Limit: 40, LRP Cluster Limit: 4) produced on



a) Configuration Comparison with Global



b) Hidden Layer Sizes

Figure 6.11: Worst and best performing networks for “DCT Both Single”

average 173.678 nodes in the autoencoder, which is near the mean of 176.926. The lowest average number of nodes produced by "DCT Both Single" was 156.628 and the highest average number was 202.702.

6.6 Discussion and Contributions

Neural networks exemplify a model of computation in which the trained, feature detectors identify the most important features of the data. Our process provides a step towards creating transferable feature detectors, represented as collections of nodes, that may be used as starting points for newly initialized neural networks. By using subsets of the layers throughout the network, our approach differs from traditional approaches that use entire trained models or complete layers as transferable components. In the fixed pre-training procedure from Chapter 5, transferring trained sets of individual hidden nodes in the network resulted in faster convergence. In this Chapter we saw that transferring the trained node subsets resulted in greater predictive accuracy as well.

In this chapter we contributed a method for changing the neural network architecture during the training process to adapt to the inclusion of data for which the initial networks were not trained. When transferable subsets of hidden nodes are identified, there are choices to be made. The nodes in the networks corresponding to the clusters being merged may have "relevance intersections," or more simply, nodes that occupy the same position in either network that have significantly different layerwise relevance propagation (LRP) values. Take, for example, when two of the spatial clusters are being merged, the networks corresponding to both clusters may have different nodes that are deemed important; however, some of these nodes may hold the same position in each network, thus causing an intersection. For example, suppose we are merging spatial clusters A and B, where nodes 3, 7, and 9 are relevant in the network corresponding to cluster A, and nodes 7, 9, and 12 are relevant in the network corresponding to cluster B. There is an intersection of nodes 7 and 9. Suppose that node 7 has approximately the same LRP value in both networks. In this case we transfer that node to the new network. On the other hand, if the relevance value for node 9 is significantly different for both networks, then we add both of these nodes to the new architecture. We call this "node splitting." In the worst case this will cause the size of each hidden layer to double upon each merge. The benefit of using this procedure, however, is that we have a way of combining multiple competing relevant nodes when combining the networks.

Our locality-sensitive clustering procedure is another contribution. This follows the line of thinking from the third scenario, which concentrates on clusters of locations for application, which was discussed in Section 4.5. This procedure not only merges spatial regions, but also merges the networks that have been trained to make predictions for data that fall within those regions. The method can be viewed as a type of "conceptual clustering" in that it satisfies the two problems that need to be addressed by conceptual clustering systems [30]. These problems are 1) determining useful subsets of an object set, which in this case is a collection of locations, and 2) determining useful concepts for each object class, which in this case are the subsets of the object set. The first problem is satisfied by dividing the data space. The second problem is satisfied by combining the relevant nodes in both neural networks corresponding to the merged clusters. Our method of locality-sensitive clustering focuses on the first problem, and we leave addressing the second for future work.

6.7 Conclusions and Future Work

From our results, we conclude that our procedure is a reasonable one to employ if the goal of the neural network is accuracy above training efficiency. Performance was indeed enhanced using our approach, but the cost of using this procedure is certainly a factor, as it is indeed a more lengthy procedure that complicates the training. Perhaps this subdivision via clustering of the dataspace is appropriate, especially in situations where one is constrained to neural networks where the expressiveness is constrained. Also, natural divisions in the dataspace may be sought, and this is one way of obtaining them.

In merging networks where there are “relevance intersections” there is a question as to whether or not it is beneficial to add the intersecting nodes from both networks. In order to control the expansion of the resulting hidden layer, further evaluation of the tradeoffs of including both nodes or of including one node and not the other should be conducted. There is a danger that just including both intersecting nodes from both networks will eventually lead to an exponential explosion in the size of the network. This increases, not only computational burden, but the risk of the resulting models overfitting in the target cluster. If we are to re-train the resulting autoencoder, this would motivate including a regularizer. Also, the online expansion of the hidden layer sizes, such as in our node splitting procedure, introduces the question of whether it is possible to use an analogous procedure for “node coalescence” to reduce the complexity of the hidden layers in another way. This is analogous to the concept (object class) merging process in COBWEB [COBWEB paper].

To determine the optimal number of clusters, our locality-sensitive hierarchical agglomerative clustering algorithm uses the Calinski-Harabasz measure. This requires the system to produce several different clusterings so that the measure can be performed and the optimal clustering selected. If we approach this clustering in a similar way to the COBWEB algorithm, the pre-determination of the number of clusters in the clustering may not be necessary. COBWEB [30] applies four operators: 1) classifying objects with respect to existing classes, 2) creating new classes, 3) combining two classes into a single class, and 4) dividing a class into several classes. In classifying objects with respect to existing classes, which we do in the case of merging clusters, locality will still have to be a factor. When performing the location clustering, each element (or location) starts out as its own class, so no new classes are created. Our procedure already pays attention to combining two classes (in this case, locations or clusters of locations) into a single class. The division of a cluster of locations into several clusters of locations is something that will be new to our procedure. In the case that merging two classes, and therefore two networks, may cause the network to no longer be representative of a portion of the new class, a mechanism for division should be created.

Finally, since we are causing the size of the hidden layers to expand through the process of discovering the relevance of certain nodes, we are in effect forcing the expressiveness of the resulting models to adapt to the conditions of the space. An interesting investigation would be to study the behavior of the networks concerning underfitting and overfitting as these expansions and contractions take place. This motivates the idea of searching for "appropriate expressiveness" of a model without generating new models, thus losing all of the intelligence captured in the training of a previous generation of networks.

CHAPTER SEVEN

CROSS-DATASET TRANSFER

7.1 Introduction

In the case of hurricanes it is easy to understand that we do not have the benefit of being able to train a DNN during the entire lifecycle of an active hurricane to make predictions about it. The real benefit of transfer learning here comes if there is the possibility of using knowledge gained from the behavior of one hurricane to predict or characterize the behavior of another. This assumes that there is enough commonality in the general behavior of these storms for the transfer to be of any use. The knowledge gained may be understood as dynamics that are fundamental to the behavior of generalized storms, but underlying conditions of terrain, solar radiation reception, diurnal cycle variations, and other variables may be quite different.

As it becomes apparent that it is possible to build up a knowledge base in the form of trained neural networks, their benefit to the task of hurricane prediction may become greater as more storms are folded into the pre-conditioning of the neural network knowledge base. This knowledge base may take the form of a database of encodings, or subsets of nodes, whose inclusion in new generations of neural networks enhances the accuracy of these models *a priori* to the new storm. To do this, we extend the research of the previous chapter to cross storm transfer and a nonparametric technique of finding encodings in trained networks.

The research detailed in this chapter builds directly on the research in Chapter 6, with three major changes.

1. We use the procedure detailed in Chapter 6 on two new storms, Harvey and Irma, using the best configuration from that analysis as a baseline. The storms have a different set of dimensions than that from the Hurricane Sandy dataset.

2. We add a new technique for transfer neuron (encoding) selection. This technique uses the Kruskal-Wallis one-way ANOVA test for independence in the selection of transferrable neurons.

3. We use networks trained on Hurricane Harvey to make predictions about Hurricane Irma and vice-versa, performing another type of transfer learning.

The data that we use in this chapter span five days at a temporal resolution of 15 minutes instead of one day at the same temporal resolution, as in the previous chapters.

This chapter is organized as follows: We begin with a background some background information about transfer learning. Following this, we discuss the data that we are using for this phase of the research, paying attention to the dimensionality of the data as well as the spatial extent. We then describe our approach, beginning with the Kruskal-Wallis One Way analysis of variance (ANOVA) method, proceeding with our method of cross-dataset transfer, and ending with a description of our general process. After this we describe the experiments that were run. We end this chapter with some conclusions and directions that future work might take.

7.2 Background

7.2.1 Transfer Learning

We address the problem of wind vector determination and prediction for hurricanes using autoencoders that have been pre-trained using unsupervised pre-training in our published work [58–60] and in the chapters that precede this one.

In this chapter we perform heterogeneous transfer learning as defined by Yang in [108]. This is defined as transfer learning where the learning effort aims to improve the learning of the target predictive function where the source and target domains share no data.

Addressing the problem of transfer learning in wind vector prediction, Qureshi, et al. used the idea of adaptive transfer learning to transfer knowledge gained from training a deep neural network from region to region to predict wind power yield in wind farms [71, 72]. The knowledge transferred was in the form of a series of autoencoders that were pre-trained, which were then fine-tuned for adaptation to the target region. Their conclusion was that adaptive transfer learning could make this model adaptable to real-time applications. They assert that what makes their concept "transfer learning" is that the underlying terrain differences from region to region change the domain sufficiently to make the task fundamentally different.

7.3 Data

The data we used in this phase of our work consists of two separate hurricanes: Harvey and Irma. Along with Hurricane Maria, these storms are known in the insurance industry as the HIM (Harvey, Irma, Maria) event of 2017. It is famous in that industry because it was a rare grouping of three major storms in a very short period of time.

Table 7.1 shows the dimensions that were collected for each data point. These dimensions differ from the dimensions in the previous chapters, mostly with respect to how the precipitation dimensions are expressed. For example, the Ice Mixing Ratio is defined as, "the ratio of the mass of ice per unit mass of dry air in a sample containing cloud ice and/or frozen precipitation [11]." The rest of the mixing ratios are defined

Table 7.1: New Data Dimensions for Each Point of Interest in the WRF Interpolated Datasets

Reading Source	**Reading Name**
WRF Radiometry Data	Temperature
	Water Vapor Mixing Ratio
	Cloud Water Mixing Ratio
	Rain Water Mixing Ratio
	Ice Mixing Ratio
	Snow Mixing Ratio
	Graupel Mixing Ratio
	Cloud Fraction
WRF Wind Data	Wind U
	Wind V
	Wind W

analogously. We also do not have the benefit of having barometric pressure as a dimension for this set of experiments.

7.3.1 Hurricane Harvey

In August of 2017 Hurricane Harvey started as a tropical wave off of the coast of Africa. It then increased in strength and extent as it moved across the Atlantic ocean, over the Caribbean, and then into the Gulf of Mexico. It weakened into a tropical storm as it crossed the Yucatan Peninsula on August 22. After this it then re-strengthened into a hurricane, making landfall on August 25th. It is responsible for 68 deaths and damage estimated at $125 billion [79]. A satellite image of Hurricane Harvey from GOES-16 just before making landfall is shown in Figure 7.1.

The data we used for Hurricane Harvey was a 1,000 kilometer square centered at 92.5 degrees West longitude and 22.5 degrees North latitude. Harvey crossed over this point over the days of August 21-25, 2017. As mentioned, these are critical days for this particular storm, as the data cover the re-intensification of the Hurricane as well as its landfall.

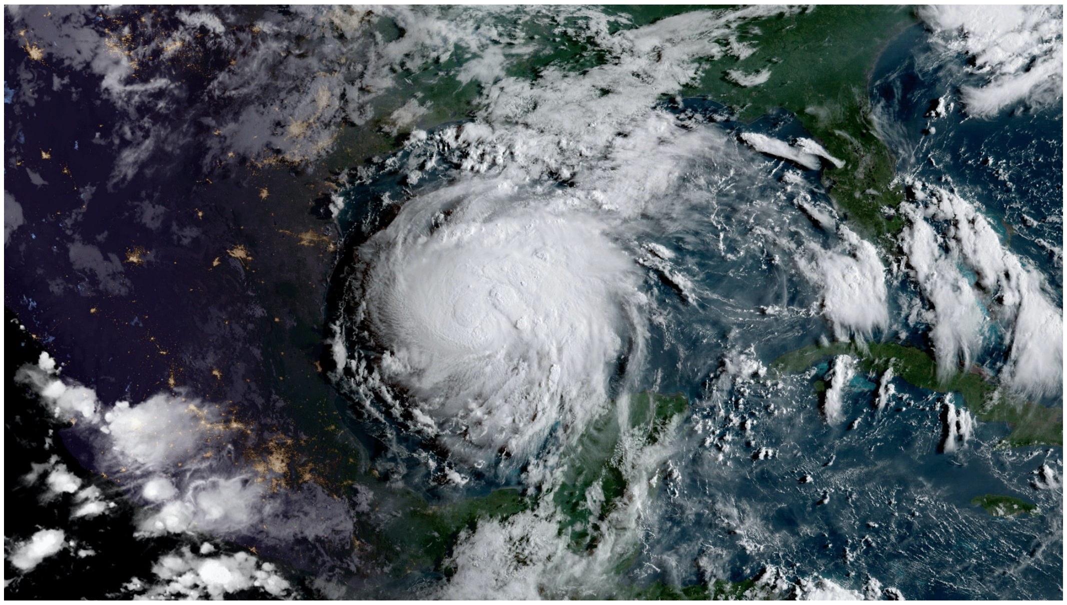


Figure 7.1: Tropical Storm Harvey on August 24, 2017 [5]

7.3.2 Hurricane Irma

Not much later in the season, just as in the case with Hurricane Harvey, Hurricane Irma started as an African easterly wave in early late August, showing convective appearance on satellite imagery on the 26th. It was classified as a category 3 hurricane on August 31. It became a category 5 hurricane on September 5th with maximum sustained winds of 185 mph, the strongest hurricane ever recorded in the Atlantic Basin outside of the Gulf of Mexico and the Caribbean. It is responsible for 47 deaths and damage estimated at $50 billion [37]. A GOES-16 image of Irma devastating Puerto Rico is shown in Figure 7.2.

The data we used for Hurricane Irma was a 1,000 kilometer square centered at 62 degrees West longitude and 17.5 degrees North latitude. Irma crossed over this point over the days of September 4-8, 2017. During this time, this storm became a category 5 storm and began its extensive destruction over the Caribbean.

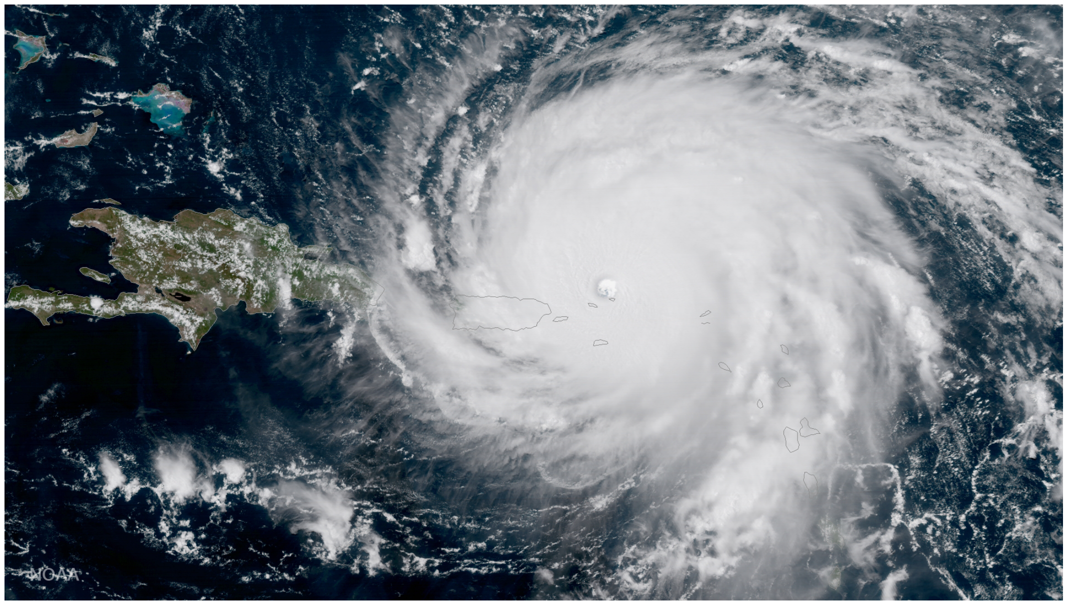
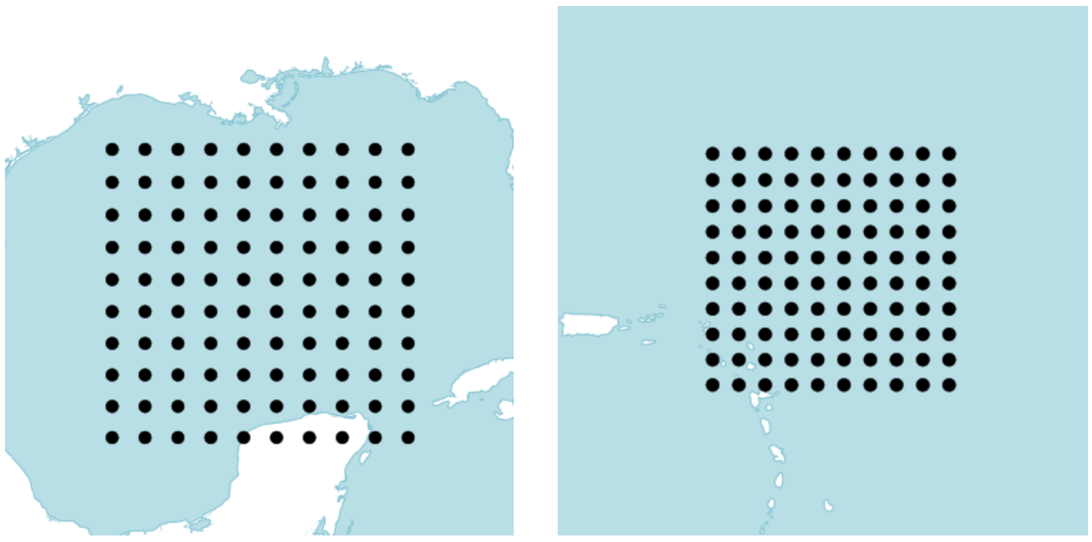


Figure 7.2: Hurricane Irma on September 6, 2017 [3]



a) Harvey area of interest b) Irma area of interest

Figure 7.3: Harvey and Irma points of interest inside respective areas of interest

7.3.3 Areas of Interest

Figure 7.3a shows the data collection points for the area of interest for Hurricane Harvey. The white area to the bottom of the figure is the Yucatan Peninsula and the Mississippi Delta (Louisiana) is at the top, situating the points of interest in the Gulf of Mexico. Figure 7.3b shows the area of interest for Hurricane Irma. It is zoomed out to make the area over which the points of interest lie easier to locate on the Earth. The small white area to the left is Puerto Rico and the small islands below the points of interest are Guadeloupe, Dominica, Martinique, St. Lucia, and other Caribbean islands.

7.4 Approach

7.4.1 Overview

In this phase of our research, we wanted to reduce the number of parameters required to run our procedure. To do this we leveraged the Kruskal-Wallis one-way ANOVA test to determine transferrability.

7.4.2 Kruskal-Wallis One Way ANOVA

The Kruskal-Wallis one-way ANOVA (KWA) method is a non-parametric way to determine whether or not samples come from the same distribution [51]. It uses the ranks of the data rather than the data itself to determine this.

The first step in performing the KWA procedure is to rank all of the data together, which means producing a collection where all of the data are assembled regardless of group membership, and ranked to produce a ranking r for each observation. The next step is to produce the test statistic H using the formula:

$$H = (N-1)\frac{\sum_{i=1}^{g} n_i(\overline{r}_{i\cdot}-\overline{r})^2}{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(\overline{r}_{ij}-\overline{r})^2}$$

Table 7.2: Example LRP values

	n_0	n_1	n_2	n_3
i_0	10	5	4	11
i_1	15	1	7	2
i_2	6	0	3	9
i_3	12	14	8	13

where n_i is the number of observations in group i, r_{ij} is the rank of observation j from group i, and N is the total number of observations across all groups. In addition:

$$\overline{r}_{i\cdot} = \frac{\sum_{j=1}^{n_i} \overline{r}_{ij}}{n_i}$$

where $\overline{r}_{i\cdot}$ is the average rank of all observations in group i. Also:

$$\overline{r} = \tfrac{1}{2}(N+1)$$

where $\overline{r}$ is the average of all r_{ij}. In a situation where there are not ties in the data, the H-statistic becomes:

$$\begin{aligned} H &= \tfrac{12}{N(N+1)} \sum_{i=1}^{g} n_i \left(\overline{r}_i \cdot - \frac{N+1}{2}\right)^2 \\ &= \tfrac{12}{N(N+1)} \sum_{i=1}^{g} n_i \overline{r}_{i\cdot}^2 - 3N - 1 \end{aligned}$$

For our purposes we are ranking the Layerwise Relevance Propagation values (LRP) of a particular node (or neuron) against the remaining neurons in the network that has been associated with a cluster. For example, in Table 7.2, we have LRP values for four nodes n_0 through n_3, for four data instances i_0 through i_3. Again, these LRP values represent the relevance of a particular node for making the decision that the network made for the corresponding data instance. In determining which nodes are transferrable (which nodes have the greatest aggregate relevance) we perform the

Table 7.3: Example LRP values separated and sorted

n_0	6	10	12	15								
$n_{1,2,3}$	0	1	2	3	4	5	6	8	9	11	13	14

Algorithm 7.1 Kruskal-Wallis transfer node identification

```
Require: $clusterA$ is a cluster of LOI's with an associated network $netA$
Require: $clusterB$ is a cluster of LOI's with an associated network $netB$
Require: $relScores$ is an associative array where $relScores[netA]$ relevance values
    for each node and each instance in the training set
 1: function KWID($newClust, prevClusters, relScores$)
 2:     $lrpSums \leftarrow \{\}$                          $\triangleright$ {} is an associative array.
 3:     $transNodes \leftarrow []$                            $\triangleright$ [] is an array.
 4:     for $prevCluster \in prevClusters$ do
 5:         for $node \in netA$ do
 6:             $krus \leftarrow Kruskal(relScores[node], relScores[netA \setminus node])$
 7:             $stat \leftarrow chisquared(krus.pvalue, d\_free)$
 8:             if $stat < krus$ then
 9:                 $transNodes \leftarrow node$
10:             end if
11:         end for
12:     end for
13:     return $transNodes$
14: end function
```

KWA procedure by separating the node to be evaluated from the rest of the nodes and performing the sorting operation as required by KWA.

Table 7.3 shows an example. Here we have separated node n_0 from the rest and performed our sorting operation on both it and the combination of nodes n_1, n_2, and n_3. The object then becomes to determine whether or not the LRP values from n_0 and the LRP values from the combination of nodes n_1, n_2, and n_3 come from the same distribution. If we determine that this is not the case, then we determine that this node is transferrable.

Algorithm 7.1 shows the procedure we used to identify transferrable node with the Kruskal-Wallis Transfer Node Identification. The algorithm tracks the procedure detailed above at the beginning of this section closely. The application of this algorithm introduces some time complexity into our procedure. For each network produced, an additional

$$O(|n|) + O(|n| \log |n|)$$

is required for the sorting and evauation of each node, where n represents the nodes in a hidden layer of a network. This gives of a total time complexity of:

$$O(|c|)(O(|n|) + O(|n| \log |n|))$$

where c represents the set of clusters.

7.4.3 Cross Dataset Transfer

The procedure detailed in Chapter 6 produces a unique clustering for each fold, with ten folds per configuration. When applying these networks to a new dataset, since we lack the clustering information for the new dataset, and furthermore lack information about how to relate clusters to clusters, we relate the clusters in the dataset from which we performed the initial training to locations in the new dataset, forming new, sometimes non-contiguous clusters for application. We did this for both a straight application to the new dataset with no fine-tuning as well as performing fine-tuning on the new dataset before application.

7.4.4 Procedure

The following subsections detail our general procedure for cross dataset transfer. We first describe our collection procedure for the two datasets under examination: Hurricanes Harvey and Irma. We then discuss the correspondence between clusters in the “from” dataset and locations in the “to” dataset.

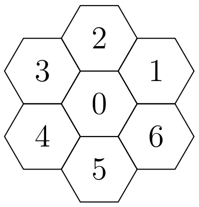


Figure 7.4: 7 cells representing one DGGS level (reproduced from Chapter 3)

7.4.4.1 Collect Corresponding Data Instances The data from Hurricane Irma was collected using the exact same procedure as the data collection for Hurricane Harvey. As with the previous iterations of our research, we collected the data using the 3D Hexagonal DGGS to bin the data into 15 kilometer hexagonal bins at three levels (above, at, and below the center cell). Figure 7.4 (Reproduced from Chapter 3) shows the concentric arrangement of these cells. We are making predictions concerning cell 0 at the center of the figure.

Our time-shifting procedure constrains the process to making predictions for the next time step (the next 15 minutes). This is in contrast to wind vector determination, or meteonowcasting, which was done in the previous chapters. In the current phase, and with the data from hurricanes Harvey and Irma, we are predicting the next time step rather than determining the wind conditions in the current time step.

7.4.4.2 Fold and Cluster Application The number of network applications to the entire new dataset (Hurricane Irma) was:

$$\text{configurations} \times \text{folds per configuration} \times \text{clusters per fold}$$

The numbers for each of these come from the application of our technique to the data from Hurricane Harvey. During the application of our procedure on that data, there were a variable number of clusters for every fold in every configuration. Lacking the data to relate the clusters from Hurricane Harvey to Hurricane Irma we elected to apply all cluster networks to the entire dataset. This was to discover any performance advantage training on a particular cluster of Hurricane Harvey has on an application to Hurricane Irma.

Each fold in the "from" configuration (Hurricane Harvey in this case) produced a unique clustering upon training, and each cluster had its own associated network that was trained to make predictions for the locations in that cluster. To relate these clusters to the new data locations, and consequently assign the cluster's network for application on these new data locations, we again used the Kruskal Wallis one-way ANOVA test to identify the degree to which the new data locations and the clusters had similar distributions. For each of the new locations, and for each of the input data variables, we selected the cluster that had the lowest mean H statistic over all of the input data variables as the cluster whose network we would use to make predictions on that location. This operation has a time complexity of $O(c \times d)$ where c is the number of clusters and d is the number of areas of interest in the new dataset, which is done in lieu of having to pre-train new autoencoders and perform the initial clustering.

7.4.4.3 Fine Tuning We tested configurations involving no fine-tuning on the data from the new dataset as well as including a fine-tuning step, as is usually performed in transfer learning. We did this to test how well the previous dataset was able to generalize to the new dataset blindly. Testing consisted of using only the testing fold that was generated at the beginning of the experiment.

Fine-tuning proceeded using training and validation dataset folds generated for the new dataset. As with the initial "from" dataset, these were time-slice folds. In other words, for each location in the new dataset, the same time-slices of data

Table 7.4: Cluster limit combinations

Data Cluster Limit:	20	30	40	50	60	70	
LRP Cluster Limit:	4	5	6	7	8	9	10

were used to test the validity of the model. For each of the clusters in the "from" configuration, the cluster's associated network was fine-tuned on the data from the locations to which it was assigned in the new dataset.

7.5 Experiments

The number and variety of experiments were dictated by the experiments in Chapter 6. However, we only used the configuration that was found to be the best in that situation: which was the "DCT Both Single" configuration.

Table 7.4 shows the cluster limits that were used. We ran all combinations of data cluster limits combined with LRP cluster limits (see Table 7.4) for both the "DCT Both Single" configuration and the KWA configuration.

7.6 Results

7.6.1 Hurricane Harvey Baseline

Figure 7.5 summarizes the performance for the Global, Both Single (30/8), and KWA (60/7) configurations. Both Single (30/8) is the configuration where the Data Cluster Limit was 30 and the LRP Cluster Limit was 8, which was found to be the best performing configuration among the Both Single Experiments. KWA (60/7) is the configuration where the Data Cluster Limit was 60 and the LRP Cluster Limit was 7, which was found to be the best performing configuration among the KWA Experiments. As can be seen, both of these configurations outperformed the Global

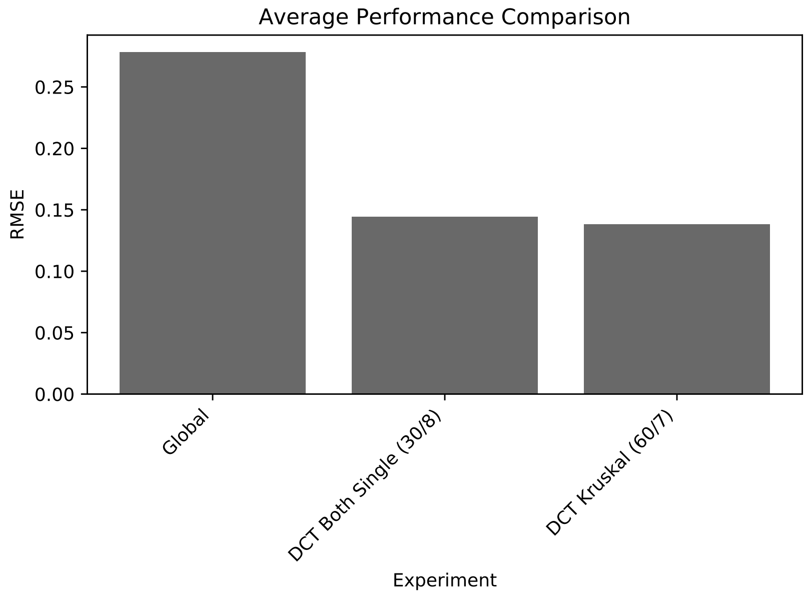


Figure 7.5: Baseline performance comparison

configuration for the Hurricane Harvey experiments, with the KWA configuration performing marginally better than the Both Single configuration.

We tested the performance of the three scenarios as we did in Chapter 6. As with the case in that chapter, at an $\alpha = 0.1$ level the two experimental configurations (Both Single (30/8) and KWA (60/7)) performed significantly better than the Global configuration. However, they did not perform significantly better at an $\alpha = 0.05$ level. Both Single (30/8) and KWA (60/7) were not significantly different from one another, even at the $\alpha = 0.1$ level.

7.6.1.1 Hurricane Harvey Global Network The Prediction Root Mean Squared Error (RMSE) for each location where the Global network was applied is summarized in Figure 7.6. As can be seen, this configuration experienced its best performance in the Northeast and Southwest of the data space. There is a contiguous area of poorer performance at the middle-South of the data space.

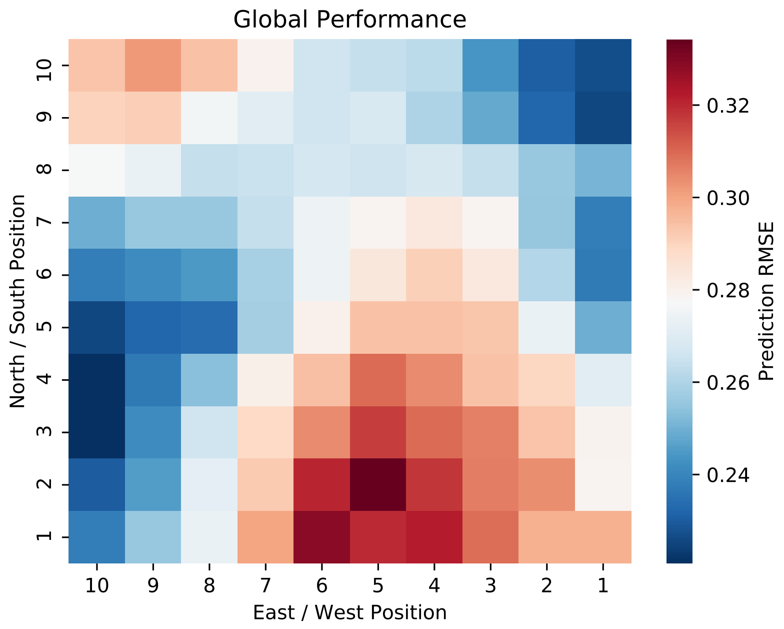


Figure 7.6: Harvey global network performance

<u>7.6.1.2 Hurricane Harvey Both Single Network</u> Figure 7.7 shows the difference in performance between each cluster configuration used for the Both Single configuration of experiments. For each cluster configuration, the figure shows the average RMSE for the Both Single configuration with the average RMSE for the Global configuration subtracted for that cluster configuration. The configuration that was shown to have performed best (Both Single (30/8)) is shown in off-white.

As explained in Chapter 6, our procedure expands the number of hidden nodes in the network. Figure 7.8 shows the results of this layer expansion for the DCT Both Single experiments. A general increase in number of hidden nodes can be seen as the Data Cluster Limit increases, with three weak modes appearing (20, 30 through 50, and 60 through 70).

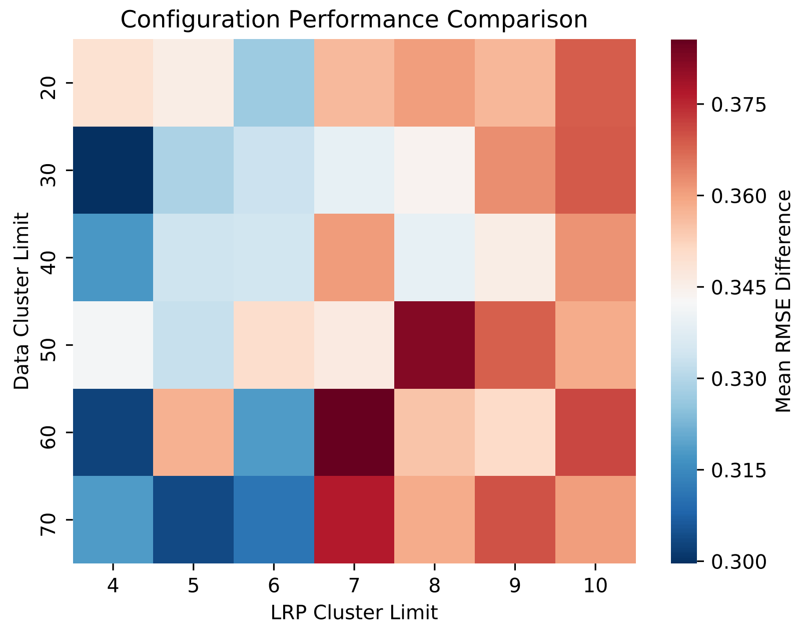


Figure 7.7: Harvey Both Single network layer performance comparison with Global

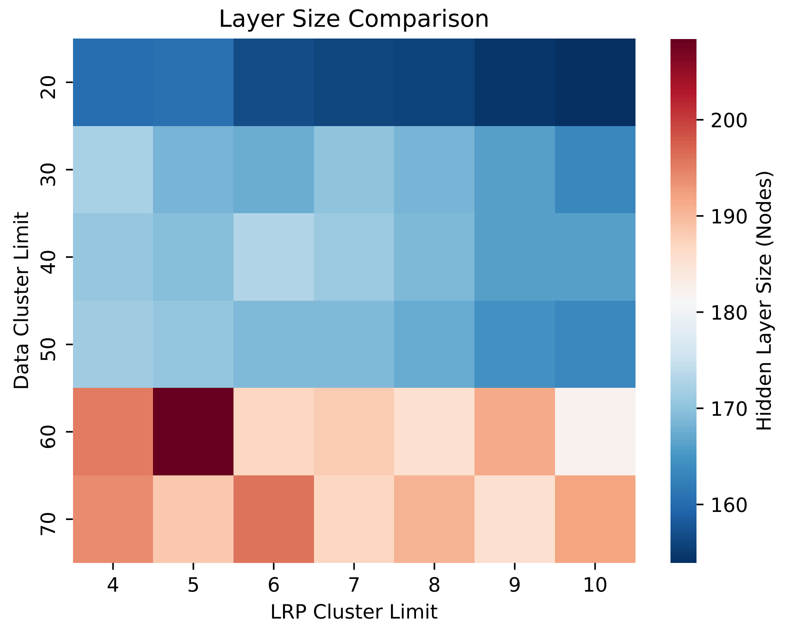


Figure 7.8: Harvey Both Single network layer size comparison

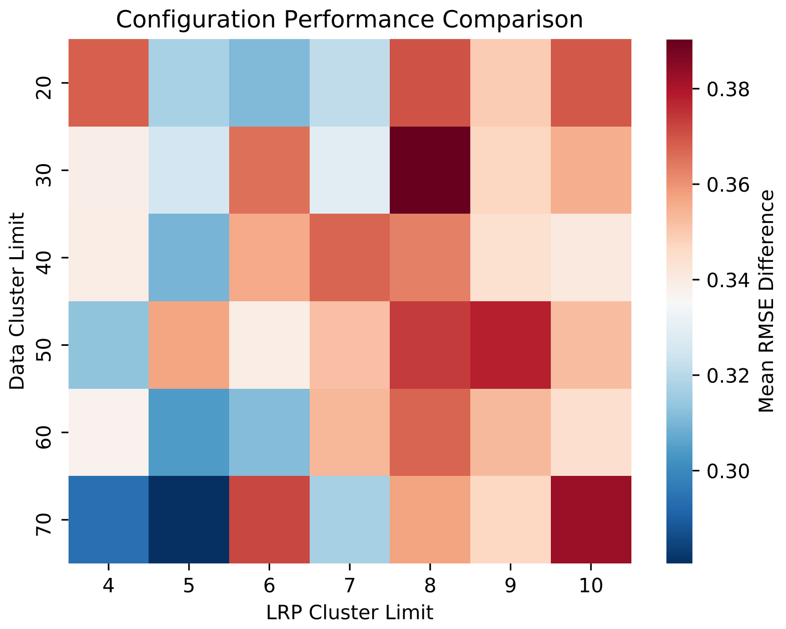


Figure 7.9: Harvey KWA network layer performance comparison with Global

7.6.2 Hurricane Harvey Kruskal Wallis One Way ANOVA

The performance comparison between the KWA configuration and the Global configuration is shown in Figure 7.9. The best performing configuration here (KWA (60/7)) is shown in the lower middle of the figure, having a difference between its performance and that of the Global configuration that is near the middle of the performance difference range. In general this range is wider than that of the Both Single (30/8) configuration difference. As shown in Figure 7.10, the KWA configuration generally produced more nodes than the Both Single configuration.

7.6.3 Hurricane Irma Baseline

A comparison to the three baselines we use for the Hurricane Irma dataset is shown in Figure 7.11. As can be seen, the Both Single and the Kruskal configurations both outperform the Global configuration on this dataset.

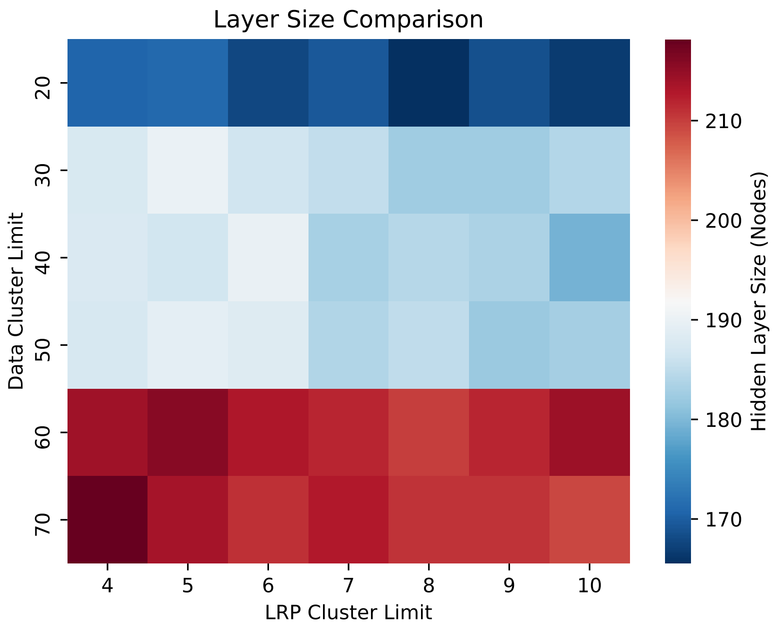


Figure 7.10: Harvey KWA network layer size comparison

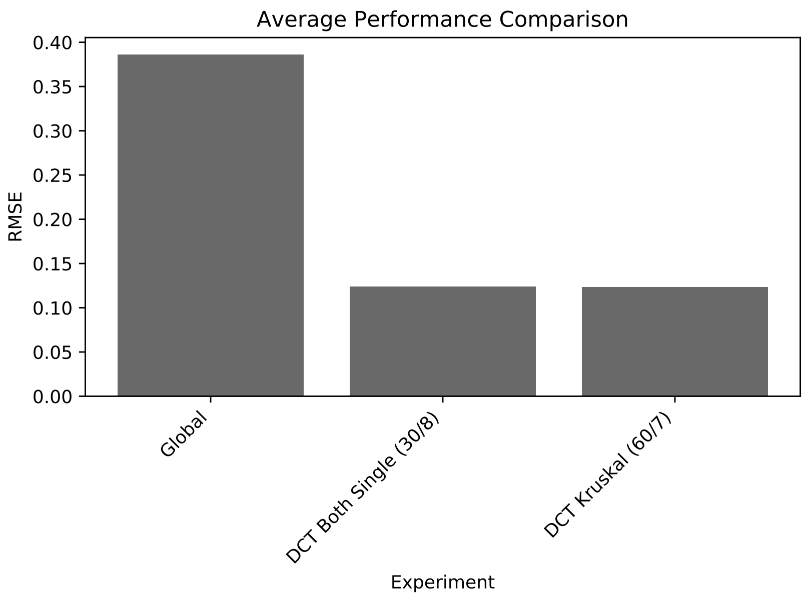


Figure 7.11: Irma baseline performance comparison

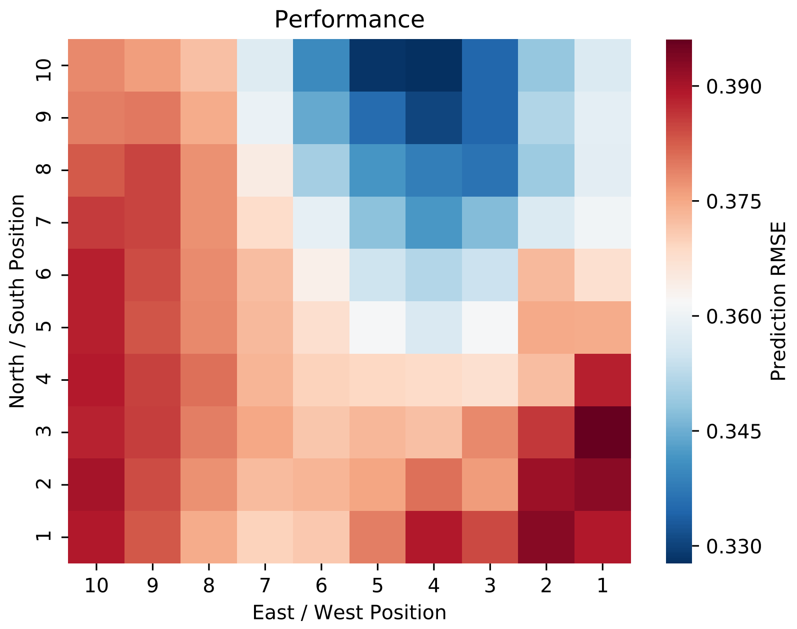


Figure 7.12: Irma Global network performance

7.6.3.1 Hurricane Irma Global Network For Hurricane Irma, the Prediction Root Mean Squared Error (RMSE) for the Global network configuration is summarized in Figure 7.12. For Irma, the best performance for this configuration was in the center-North of the AOI. The worst performance was in the extreme Southeast.

7.6.3.2 Hurricane Irma Both Single Network The heatmap showing the performance comparisons for the Both First configuration used on Hurricane Irma is shown in Figure 7.13. The best configuration in this experiment had a data cluster limit of 20 and an LRP cluster limit of 10. It is show in the extreme upper right hand corner. The layer size comparison heatmap for the Both First configuration used on Hurricane Irma is shown in Figure 7.14. As is the case with all of the experiments, greater data cluster limit models produced more nodes in the hidden layer in general.

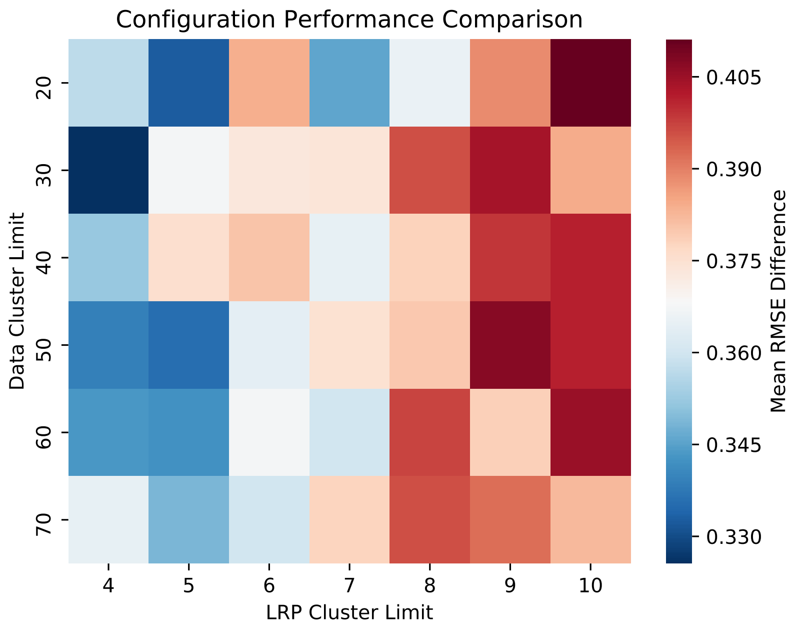


Figure 7.13: Irma Both Single network performance

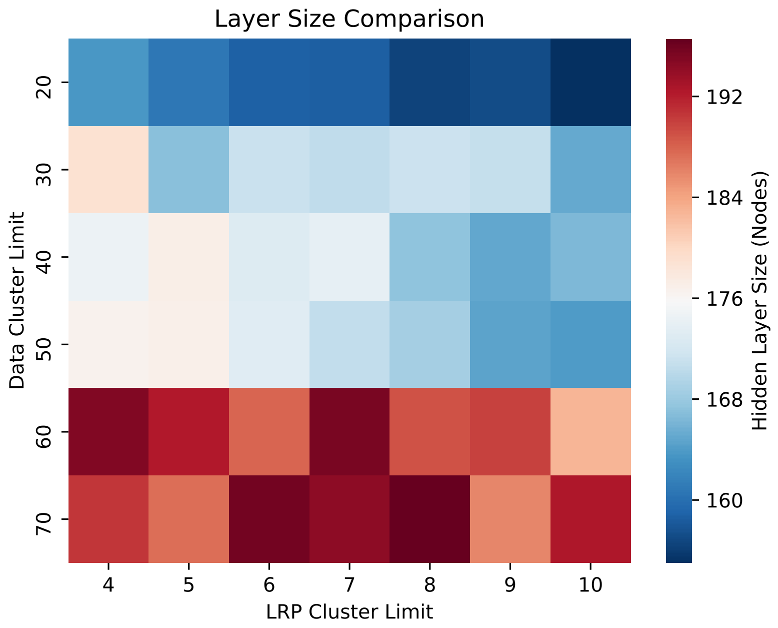


Figure 7.14: Irma Both Single network layer size comparison

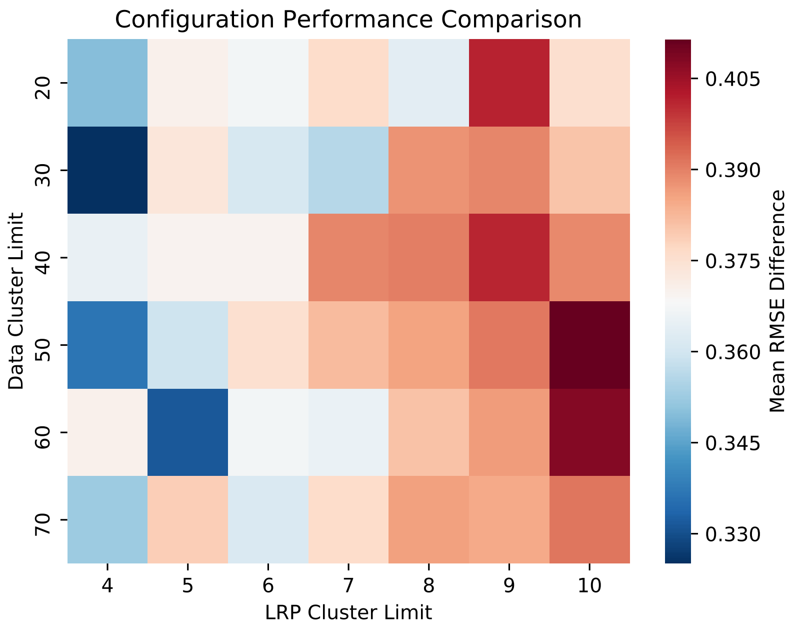


Figure 7.15: Irma KWA network performance

7.6.3.3 Hurricane Irma Kruskal Wallis One Way ANOVA The performance comparison between the KWA configuration and the Global configuration for Hurricane Irma is shown in Figure 7.15. The best performing configuration here (KWA (50/10)) is shown on the extreme right of the figure in the darkest square. As is the case with the Hurricane Harvey KWA configuration, and shown in Figure 7.16, the Hurricane Irma KWA configuration generally produced more nodes than the Both Single configuration.

7.6.4 Harvey to Irma Transfer

Our first storm-to-storm transfer experiment was transferring models that were trained on Hurricane Harvey to application on Hurricane Irma. We first studied transferring the Global network configuration as a baseline for the other experiments.

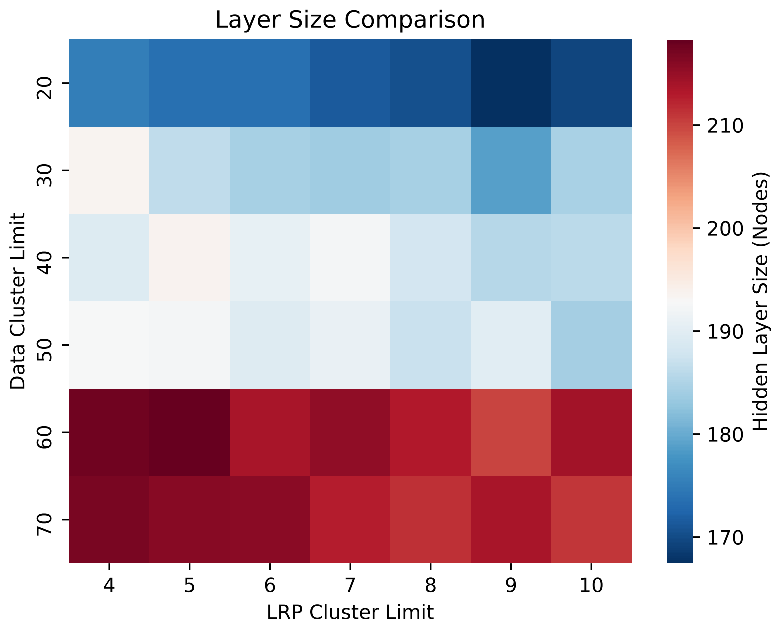
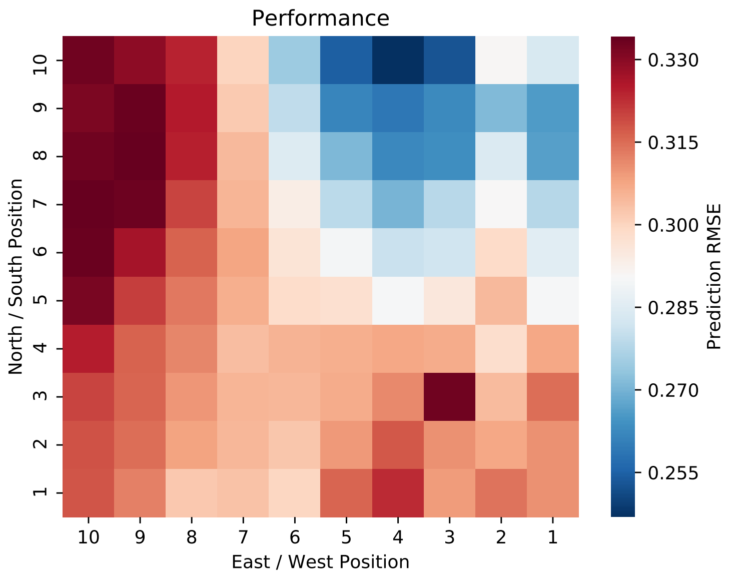


Figure 7.16: Irma KWA network layer size comparison

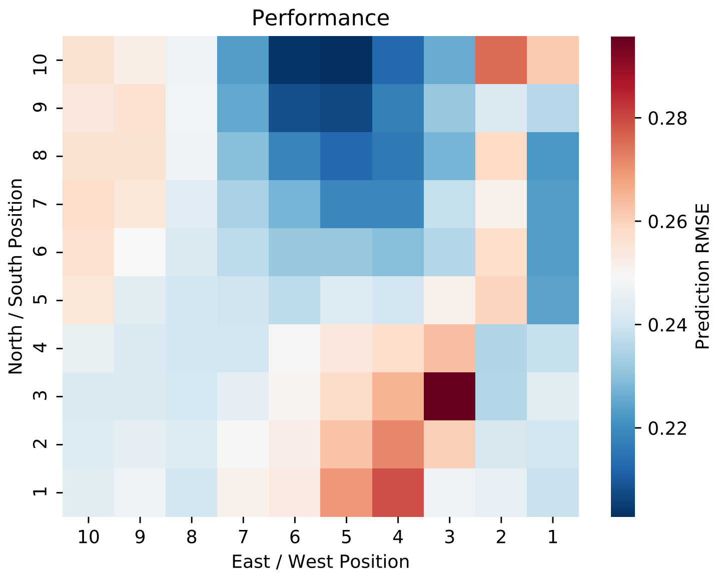
We then studied how the Both Single and KWA configurations performed when we transferred their networks.

7.6.4.1 Global Transfer When transferring the Global network that was trained using Hurricane Harvey as the "from" network, it is apparent that fine tuning made a large difference. This is shown in the contrast between Figure 7.17a and 7.17b. Figure 7.18 shows the difference between the Global network configuration without fine tuning on the Hurricane Irma dataset and with fine tuning. The figure shows that the greatest performance improvements happened in the west of the AOI.

Very interestingly, the performance of the Global network trained on the data from Hurricane Harvey that is transferred and fine-tuned for Irma is better than that of the Global network trained on and applied to the data from Hurricane Irma.



a) Global network transfer performance



b) Global network fine-tuned transfer performance

Figure 7.17: Performance of non fine-tuned vs. fine-tuned global networks trained on Hurricane Harvey and applied to Hurricane Irma

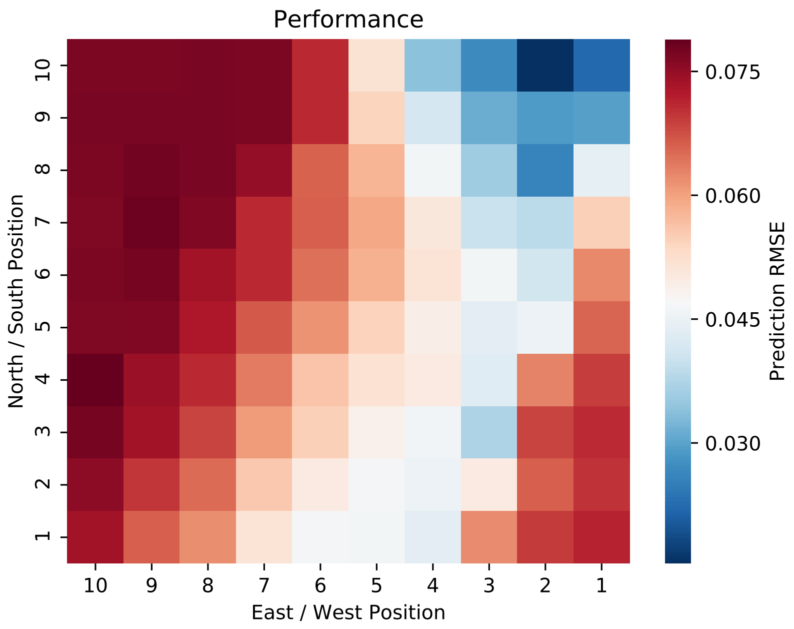
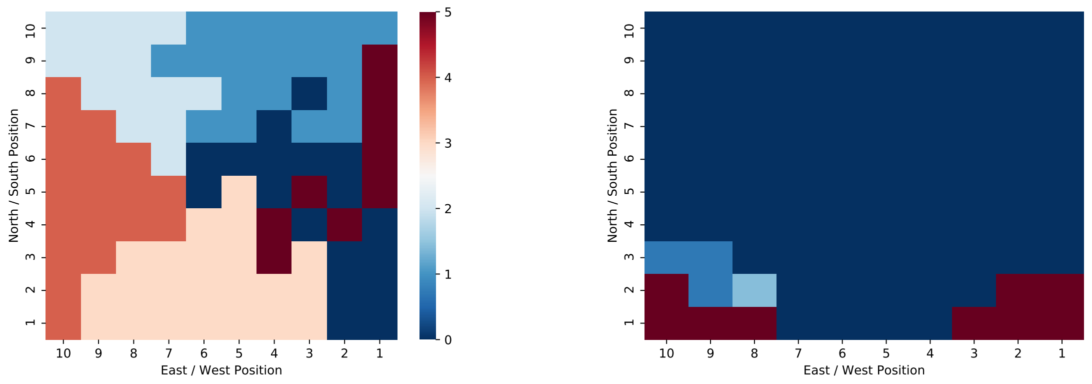


Figure 7.18: Performance difference (Harvey to Irma global network no fine tuning - fine tuning)



(a) Fold 1 clustering (b) Application from fold 1 clustering

Figure 7.19: Harvey Both Single best configuration clusterings: fold 1

7.6.4.2 Both Single Transfer Performance We begin our examination of the performance of our procedure by looking at the clusterings produced by the Both

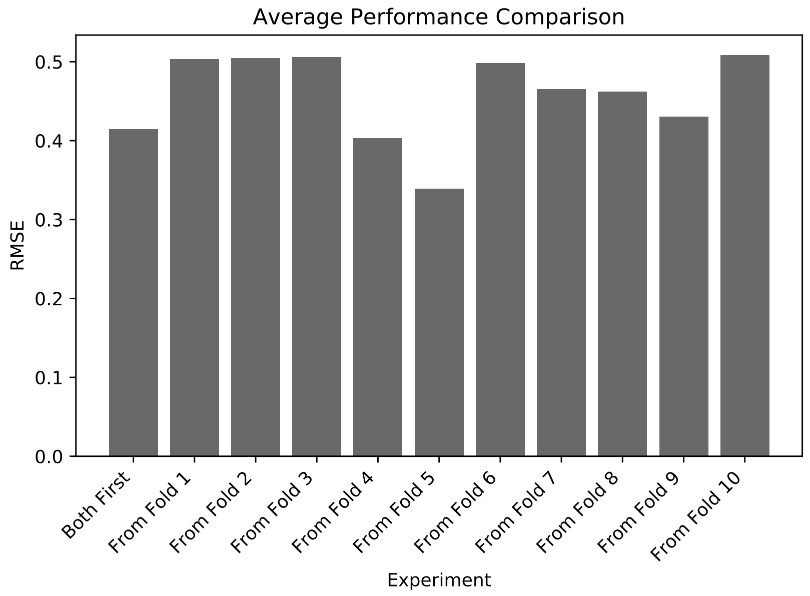


Figure 7.20: Performance comparison: Harvey to Irma Both Single

Single network configuration initially trained on Hurricane Harvey and applied to Hurricane Irma. Figure 7.19a shows the clustering that resulted from fold 1 of the Both Single best configuration (20/10) applied to Hurricane Harvey. As can be seen, the clusterings are not concave and in some cases appear disconnected due to diagonal locations being added to clusterings. Figure 7.19b shows what clusters from this initial clustering were applied to which areas of Hurricane Irma. The dark blue cluster dominates this entire figure, indicating that this data was the most similar to the corresponding dark blue cluster scattered around the Southeast of Figure 7.19a. This was typical all of the folds.

The performance comparisons for all “from folds” are shown in Figure 7.20 for the Harvey to Irma Both Single experiment that did not involve fine tuning and Figure 7.21 shows comparisons for the experiment where the networks were fine-tuned. Since the “from folds” from Hurricane Harvey were applied to 10 “to folds” this chart represents an average of the performance across all 10 “to folds” from

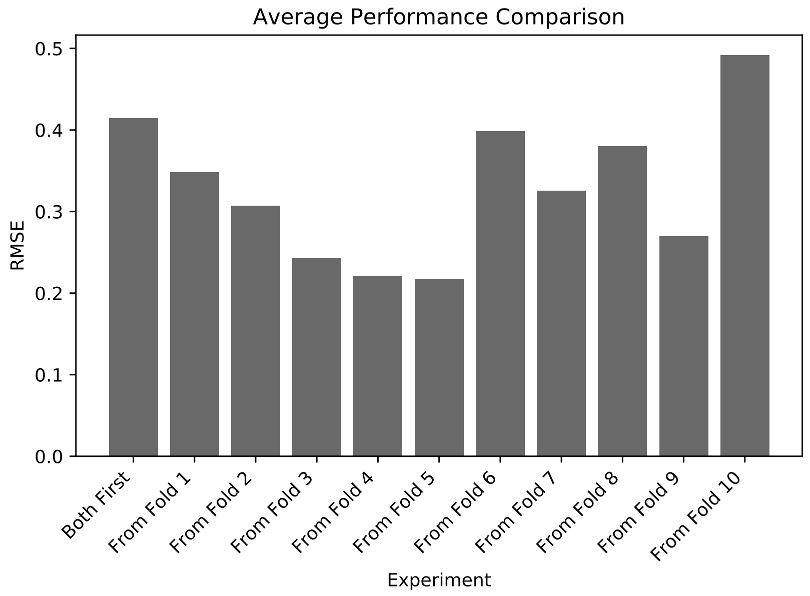


Figure 7.21: Performance comparison: Harvey to Irma Both Single fine tuned

Hurricane Irma. This is consistent across all of our experiments except for the Global configuration. The aggregate performance of the application of the networks with no fine tuning is not discernibly different from the Both First configuration. However, when fine tuning is applied the performance improves noticeably, achieving the best performance when using fold 5 on the new dataset.

Figure 7.22 shows a heatmap of the performance for the Both Single configuration for Hurricane Irma that performed best. The (20/10) in the caption indicates that this configuration had a data cluster limit of 20 and an LRP cluster limit of 10. We use this as a baseline to compare the transfer performance from Hurricane Harvey's Both Single configuration and the transfer of these networks for operation on Hurricane Irma.

For the Both Single configuration Figure 7.23 shows the performance of fold 5 of this model trained on Hurricane Harvey and transferred to Hurricane Irma. A comparison of Figure 7.23 a and b to Figure 7.22 shows that they both outperform the

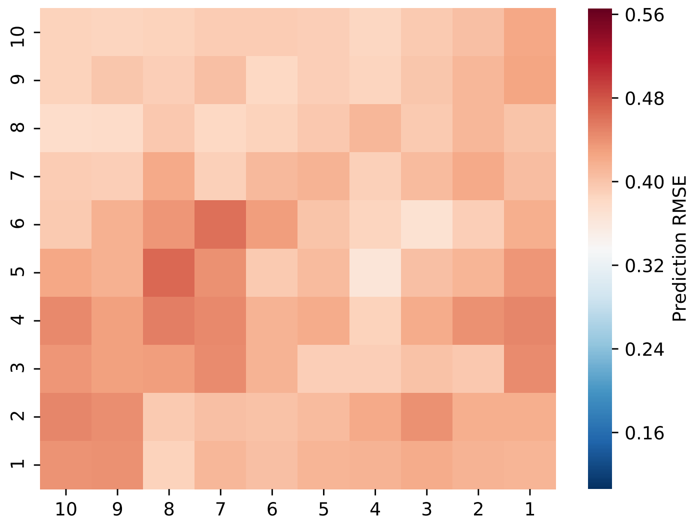
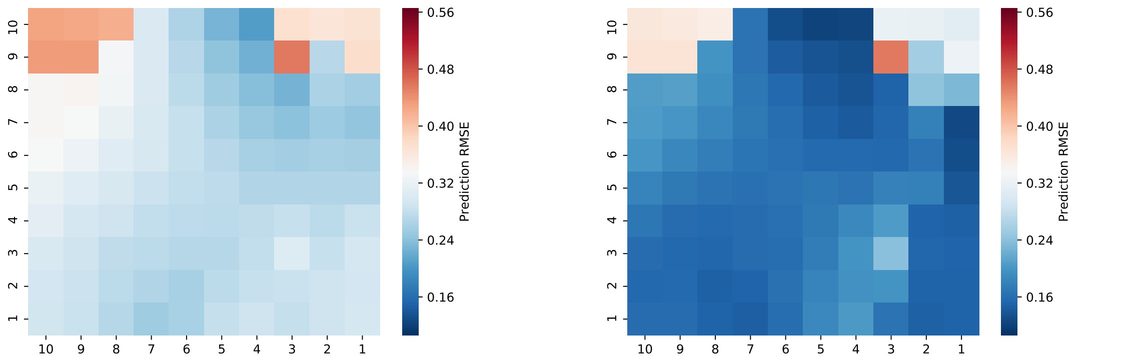


Figure 7.22: Irma Both Single best performance (20/10)

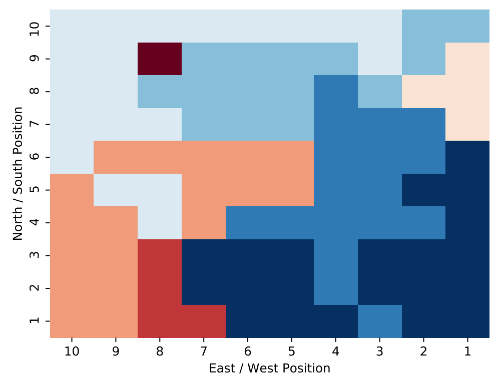


(a) Both single from fold 5

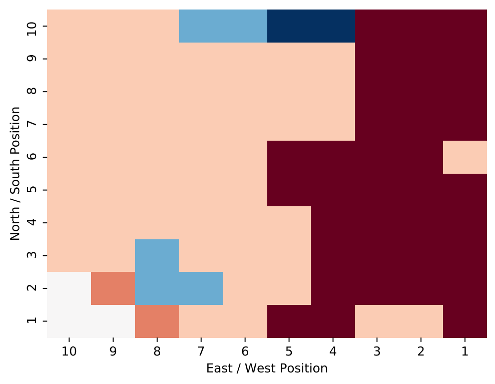
(b) Both Single Fine-Tuned From Fold 5

Figure 7.23: Harvey to Irma Both Single (non-fine-funed vs. fine-tuned) Fold 5

Both Single (20/10) configuration that was trained on Hurricane Irma. A comparison between Figure 7.23a and Figure 7.23b shows the improvement that fine tuning confers in all but one location of the space in the Northeast.



(a) Fold 9 clustering



(b) Application from fold 9 clustering

Figure 7.24: Irma KWA best configuration clusterings

<u>7.6.4.3 Kruskal Wallis one-way ANOVA (KWA) Transfer Performance</u> The second set of transfer experiments we ran was the Kruskal Wallis one-way ANOVA (KWA) configuration, transferring the networks from training on Hurricane Harvey to application on Hurricane Irma. The clustering that resulted from training the networks using this configuration on Hurricane Harvey is shown in Figure 7.24a. An application of this clustering to the data from Hurricane Irma, shown in Figure 7.24b, shows that this situation is not quite as dominated by the application of a single cluster's network. It also shows that networks that were trained on very small areas of Harvey's dataspace were applied to much larger areas of Irma's dataspace. For example, the small peach colored area in the Northeast ended up being most similar to most of the data in the Hurricane Irma dataset.

It is apparent in this case, as it was in the case of transferring the Both Single configuration from Harvey to Irma, that fine-tuning offered performance improvements for at least half of the transfer experiments. This is shown in the

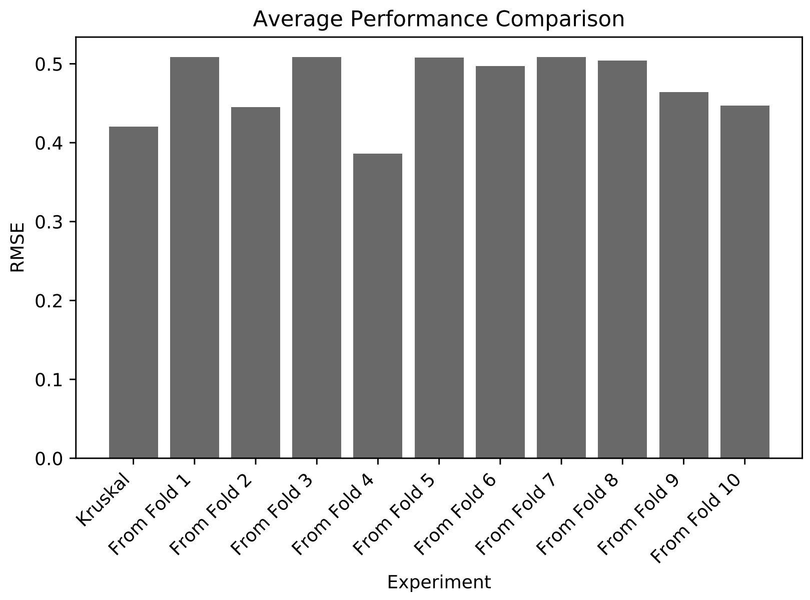


Figure 7.25: Performance comparison: Harvey to Irma KWA

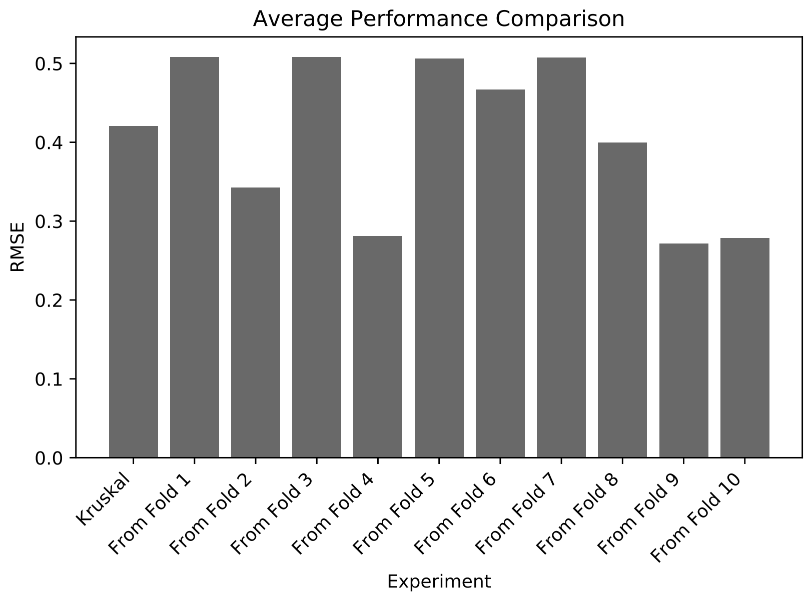


Figure 7.26: Performance comparison: Harvey to Irma KWA Fine Tuned

contrast between Figure 7.25 and Figure 7.26. The greatest performance improvement was in the case of fold 9.

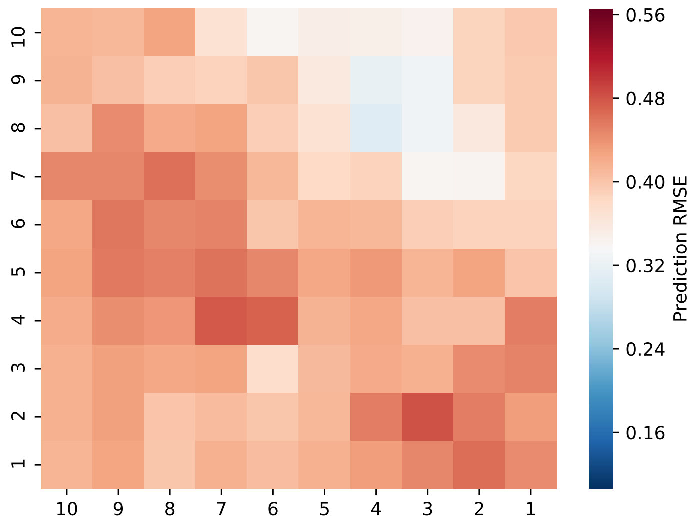
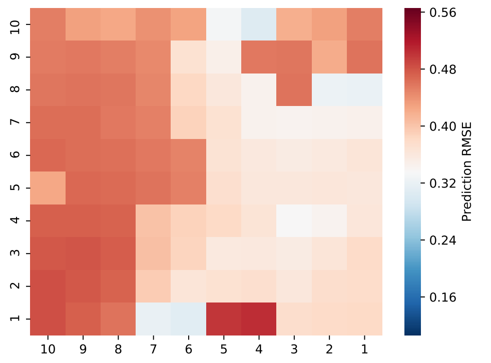
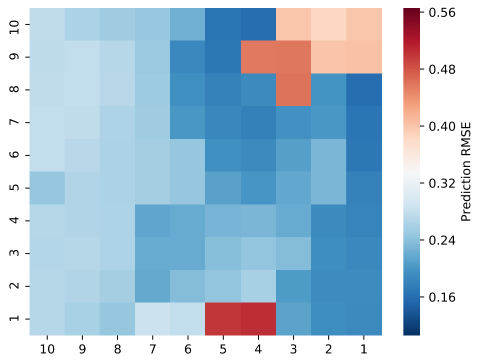


Figure 7.27: Irma KWA best performance (50/10)



(c) KWA From Fold 9



(d) KWA Fine-Tuned From Fold 9

Figure 7.28: Harvey to Irma KWA (non-fine-tuned vs. fine-tuned) folds 1 through 3

Comparing the best performance of the KWA configuration, shown in Figure 7.27, to the best performing transfer configurations in Figure 7.28 we can see that there was no performance gain in the non-fine-tuned KWA transfer (Figure 7.27a). But as with the Both Single configuration, a performance advantage for the majority of locations in the Hurricane Irma data was realized by applying fine tuning, as seen in Figure 7.27b.

7.6.5 Irma to Harvey Transfer

Having studied transferring our models that were trained on the data from Hurricane Harvey to application on the data from Hurricane Irma, we also conducted experiments in the opposite direction. This section describes the performance when we applied our technique from training on Hurricane Irma to application on Hurricane Harvey.

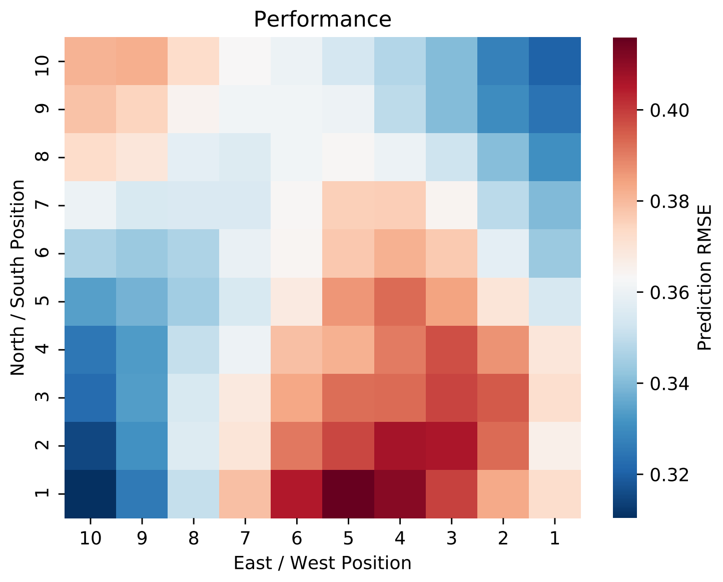
7.6.5.1 Global Transfer Performance

As was the case with transferring the global network configuration from Harvey to Irma, there was a performance advantage when fine tuning upon transferring this configuration from Irma to Harvey. Though the diagrams appear similar, Figures 7.29 a and b show this performance improvement. The performance differences appear in Figure 7.30, which shows the fine-tuned performance subtracted from the fine-tuned performance.

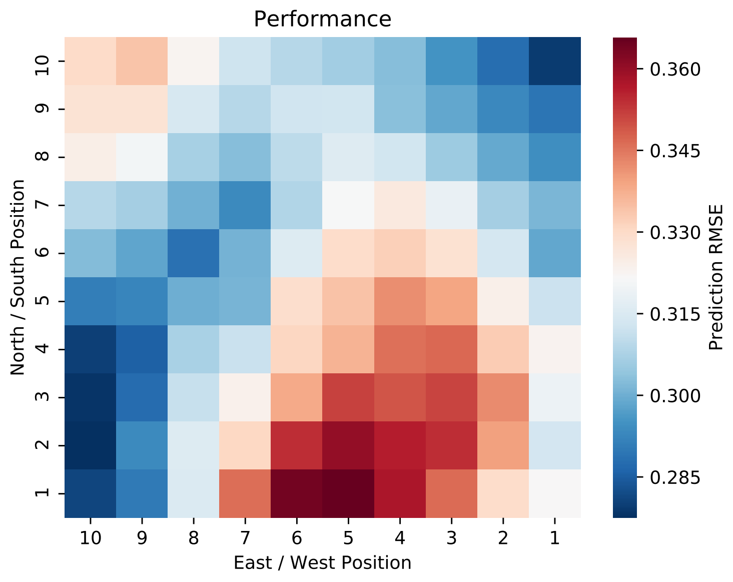
7.6.5.2 Both Single Transfer Performance

In fold 9, the data from Hurricane Harvey was found to be most similar to 14 locations located near the middle of the AOI. This is shown in Figures 7.31 a and b. The figure also shows that only a subset of the clusters available were applied from Irma to Harvey.

The Both Single configuration trained on Irma and applied to Harvey experienced a greater apparent performance improvement when fine tuning was applied. This is illustrated in the comparison of Figures 7.32 and 7.33. With the exception of fold 1, all of the folds performed better than the Both First baseline when fine-tuning was applied.



a) Global network transfer performance



b) Global network fine-tuned transfer performance

Figure 7.29: Performance of non fine-tuned vs. fine-tuned Global networks trained on Hurricane Irma and applied to Hurricane Harvey

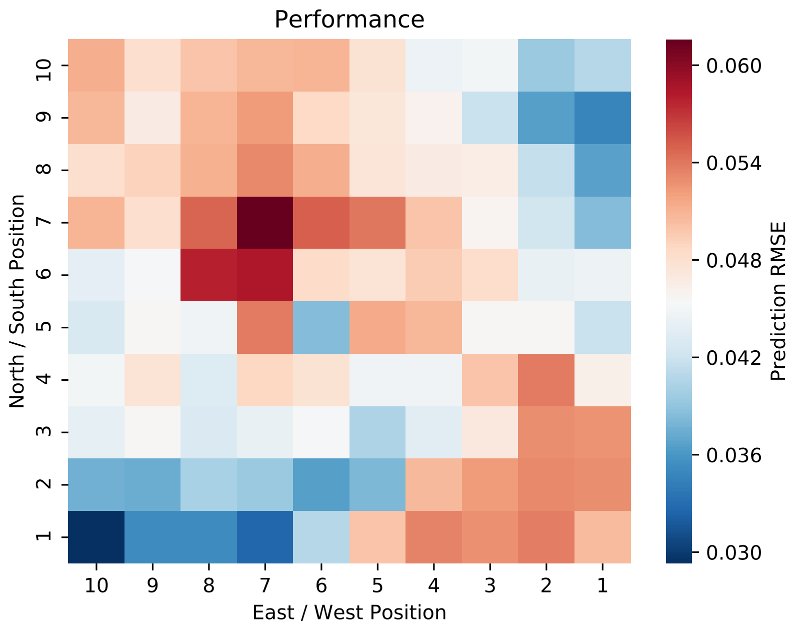
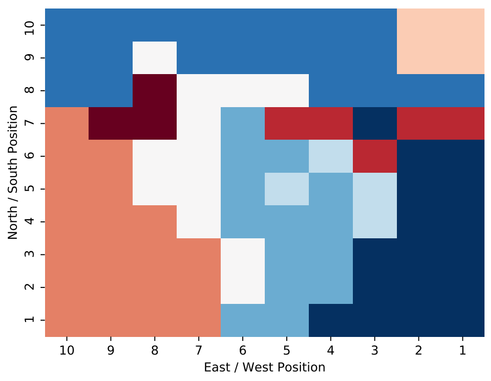
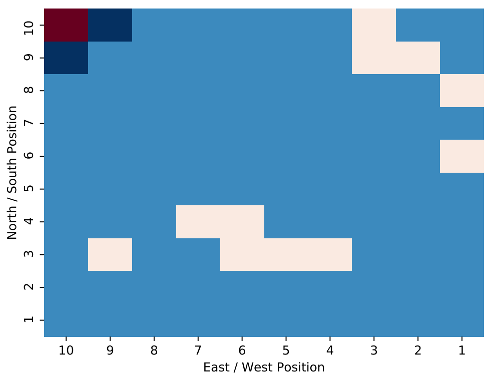


Figure 7.30: Performance difference (Irma to Harvey Global no fine tuning - fine tuning)



(a) Fold 9 Clustering



(b) Application From Fold 9 Clustering

Figure 7.31: Irma Both Single best configuration clusterings (continued)

As our baseline for comparison we included the heatmap of the performance of the best clustering from the Both Single experiment trained on Hurricane Harvey in

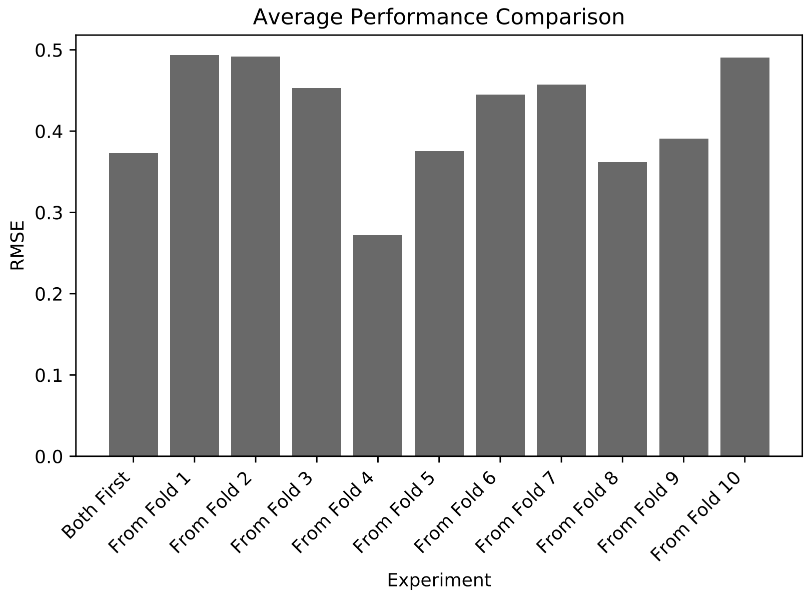


Figure 7.32: Performance comparison: Irma to Harvey Both Single

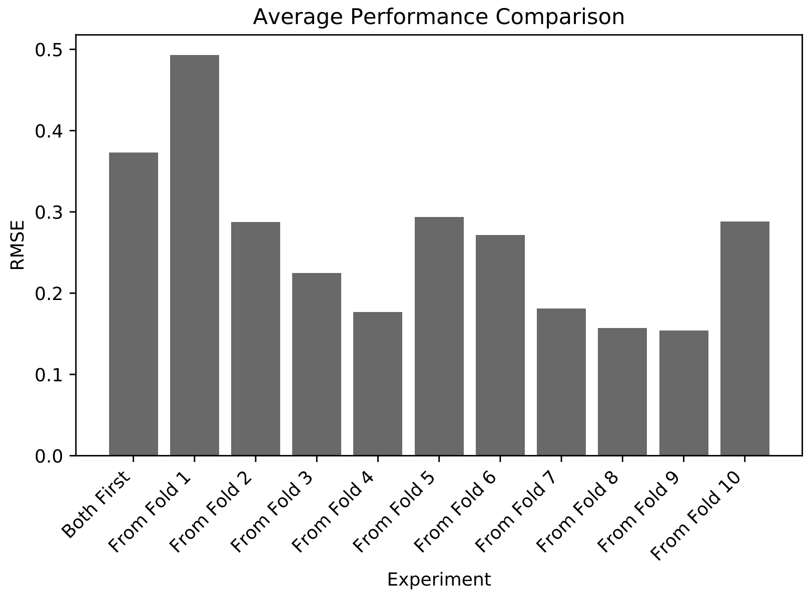


Figure 7.33: Performance comparison: Irma to Harvey Both Single Fine Tuned

Figure 7.34. The best performing fold in this experiment was fold 9, whose non-fine-

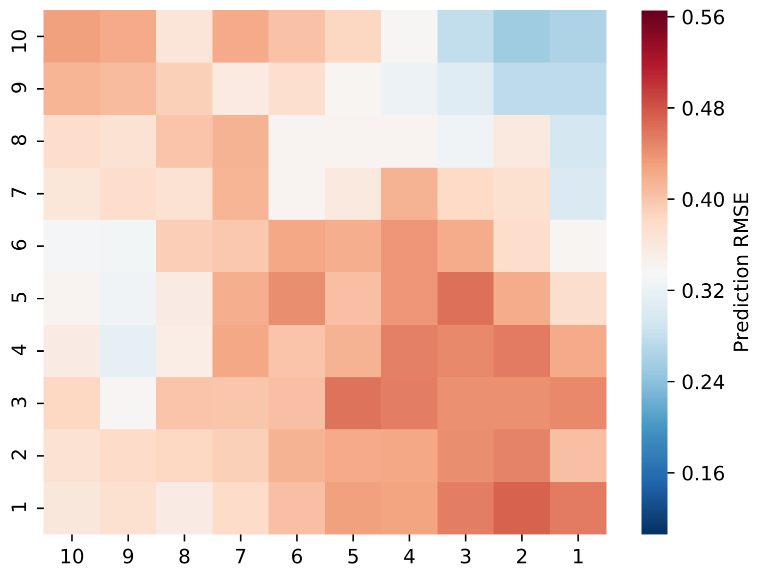
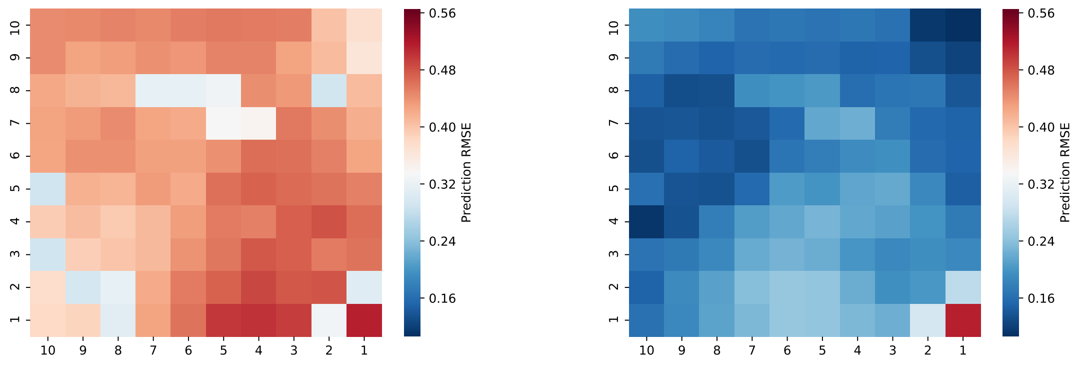


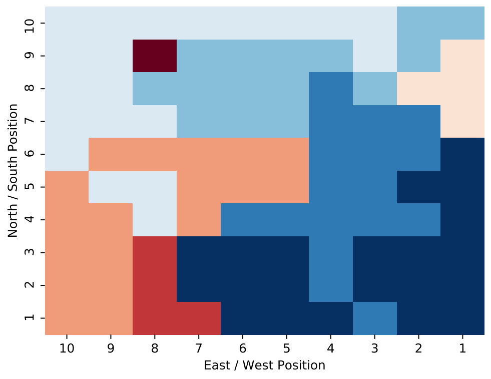
Figure 7.34: Harvey Both Single (60/7)



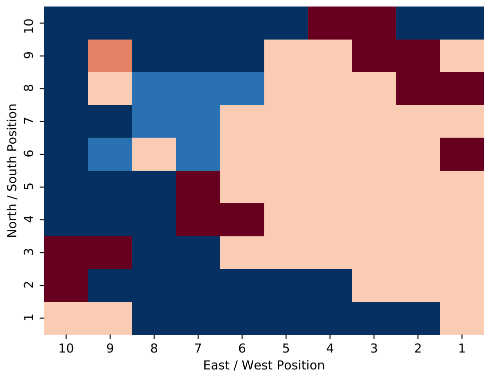
(a) Both Single From Fold 9 (b) Both Single Fine-Tuned From Fold 9

Figure 7.35: Irma to Harvey Both Single (non-fine-tuned vs. fine-tuned) Fold 9

tuned and fine-tuned performances are illustrated in Figures 7.35 a and b. There was an enormous advantage in fine tuning this configuration.



(c) Fold 9 Clustering



(d) Application From Fold 9 Clustering

Figure 7.36: Harvey KWA best configuration clusterings (continued)

7.6.5.3 KWA Transfer Performance The clustering from fold 9, both the original application of the KWA configuration on fold 9 of the original data and the application of this clustering to the Hurricane Harvey data is shown in Figure 7.36 a and b. The application map in this instance appears more evenly divided, dominated by the peach and blue clusters.

The performances of each of the "from folds" are summarized in Figures 7.37 and 7.38. It is again shown that there is an advantage to fine-tuning. The performance advantage is also shown in the comparison of the heatmaps in Figures 7.39 and 7.40.

7.7 Discussion and Contributions

In current practice, transfer learning uses models that have been trained in related domains, or with compatible input data configurations. The models are imported, fine-tuned, and deployed. We contributed a method of creating systems of transferrable models that are then transferred to other datasets compatible with the original one. By compatible, we mean that the systems of transferrable models

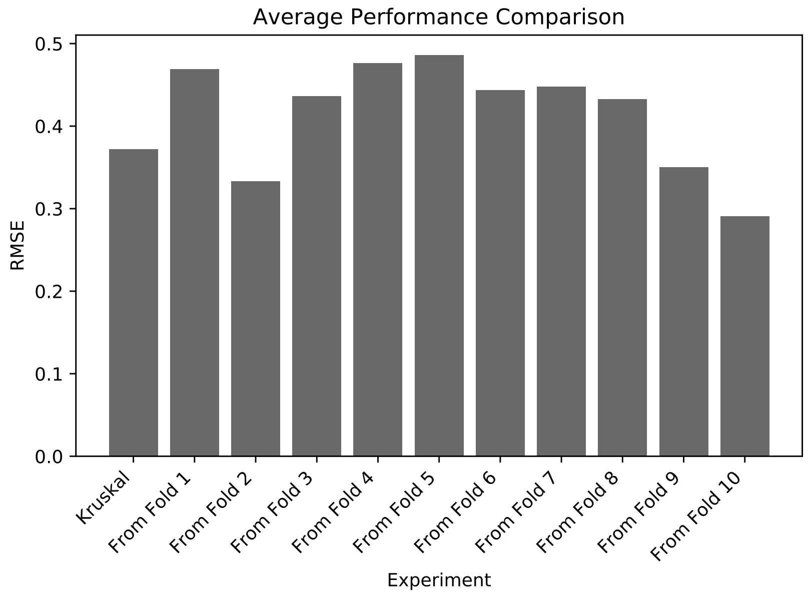


Figure 7.37: Performance comparison: Irma to Harvey KWA

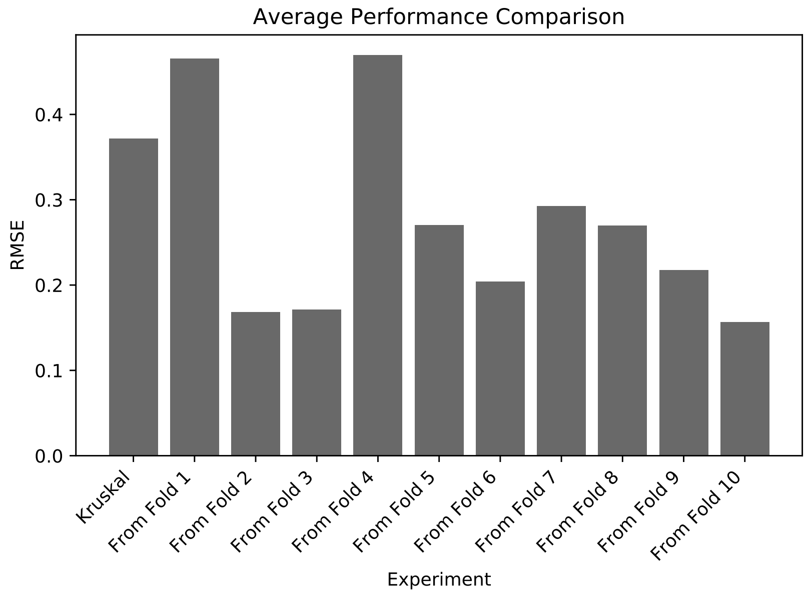


Figure 7.38: Performance comparison: Irma to Harvey KWA Fine Tuned

are meant to be transferred to domains that are related, such as from spatiotemporal model to spatiotemporal model, especially in situations where the data dimensions

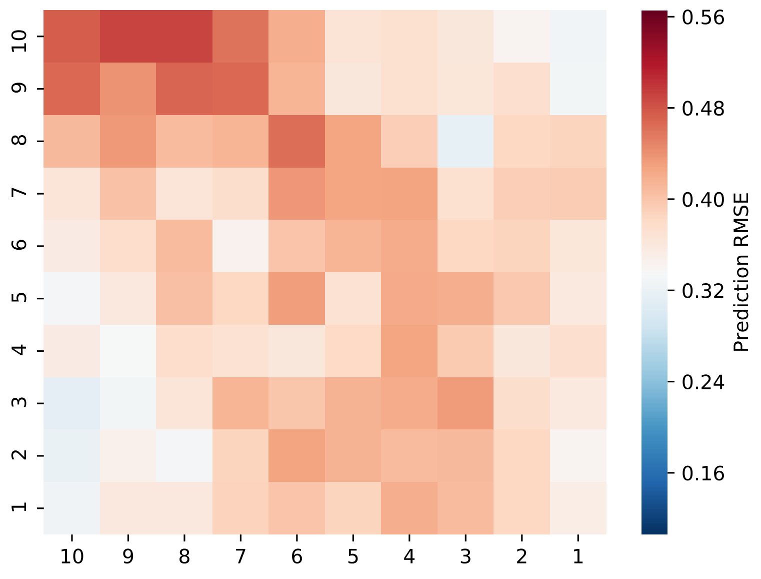
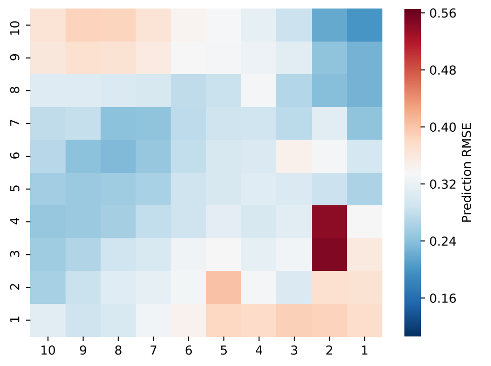
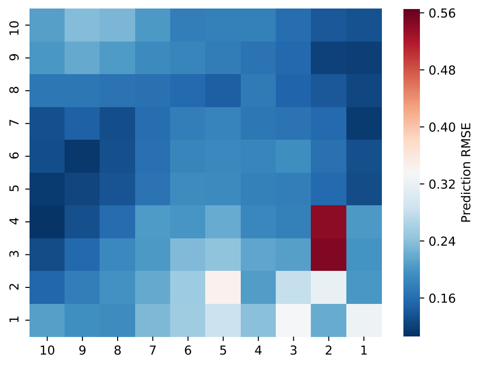


Figure 7.39: Harvey KWA (50/10)



(c) KWA From Fold 10



(d) KWA Fine-Tuned From Fold 10

Figure 7.40: Irma to Harvey KWA (non-fine-tuned vs. fine-tuned) fold 10

represent the same thing in both domains. In the transfer learning literature this is referred to as *heterogeneous, noninductive transfer learning* [108]. This means that regions of the source data space will be affected similarly to certain regions of the target data space. With proper parameterization, this removes the need to perform any pre-training on the target data space, which now only requires fine-tuning. The target data space then has the benefit of using the feature detectors from our source. In our experiments, the transfer and fine-tuning among datasets was found to yield a significant performance improvement without the need for the initial training on the dataset to which the models were transferred.

Another interesting contribution that comes as a consequence of our set of experiments is the need to define a "cluster to application mapping." This is the process where a clustering is created during the initial training of a model before transfer and to the new dataset. Each cluster in this initial clustering has an associated network that has been trained to operate on any location within that cluster. When the new dataset is being considered, for each location in this new dataset we determine the most similar cluster from the initial dataset. In other words, the primary clustering's characteristics determine which model to use in each location in the the secondary dataset. It is the network corresponding to the cluster from the first dataset that is fine-tuned and then makes predictions in the target location.

7.8 Conclusions and Future Work

The Kruskal-Wallis one-way ANOVA test is a reasonable non-parametric extension to and development of the previously used forms of encoding identification from Chapter 6. It performs at least as well as these other configurations and has the benefit of being a statistically-derived method. Whether or not it is worth the extra time in training is a question that is highly domain dependent.

However, whatever the rigor involved in training and testing the original model, it does appear that the transfer of a model from storm to storm provides some advantage in application. The effect is enhanced when fine-tuning is applied, rendering it truly a transfer learning procedure. In our studies this was more effective when applied from Irma to Harvey rather than from Harvey to Irma. The reasons for this could be the subject of studies at the nexus of computer science and meteorology.

We intend to study the properties of "cluster to application" mapping from the primary dataset (dataset upon which the networks were initially trained) to the target dataset (dataset for application) in cross dataset transfer. The "cluster to application" mapping often results in applying networks from the primary dataset to non-contiguous areas of the second dataset. At this point we do not know whether or not this affects performance negatively. To compare the clustering in the primary dataset and the "cluster to application" mapping in the target dataset, we could use the Calinski-Harabasz measure to determine the similarity of the clusterings. Then, to study how contiguity affects performance in the second dataset's locations for application, we would experiment with the pre-clustering of the second dataset before the relationships are drawn between the initial clustering and the locations in the second dataset. At a cost of $O(n^3)$ for the clustering, where n is the number of locations in the new dataset, this would cause the complexity of finding which clusters' networks from the initial dataset to apply to the locations of the second dataset to be $O(c_1 * c_2)$ where c_1 and c_2 are the clusters from the first and second datasets, respectively. If we assume that the number of datapoints in each cluster in the second dataset is greater than 1, the Kruskal-Wallis one-way ANOVA procedure to relate the clusters across data sets will need to run at most half as many times. An open question here is whether or not this will enhance aggregate accuracy by causing the areas of application in the second dataset to be less dispersed.

The use of the Kruskal-Wallis one-way ANOVA method for determining transferrability of nodes, being a step in the direction of reducing the number of parameters required to run our procedures, signals our desire to reduce the number of parameters as a whole that have to be given to the system. As of now there are many parameters beyond those governing the transferrability determination of nodes. For example, there are those governing the initial starting point for hidden layer size and whether or not to apply regularization. These are on top of the usual neural network parameterizations of epoch size, learning rate, batch sizes, etc. Automation or flexibility in this area could be a major achievement in deep learning.

CHAPTER EIGHT

CONCLUSION

8.1 Major Contributions

The encoding and transfer of knowledge in neural networks is a very important topic for study, as these models are becoming ubiquitous. In this dissertation we have taken a branch of this endeavor and, using an application to meteorology as a area for testing and exploration, contributed a technique that could be developed into a way for these models to share information and improve accuracy. We have detailed several contributions in this area. We recap the most pertinent ones here:

8.1.1 Autoencoders in Meteorology

We have demonstrated of the use of autoencoders in meteorology, particularly for the purposes of wind vector determination in three hurricane events. This was a necessary first step of our investigations, as it provided a basis for all of the other experimentation. We discovered that our base technique of using autoencoders in a manner similar to the execution of Numerical Weather Prediction (as stated in Chapter 2) yields results that tell us that there is a functional relationship between radiometric data and wind vectors. This functional relationship can be exploited in the training of neural networks, using unsupervised pre-training, for wind vector determination. In doing this, we gained the confidence to move into more analytical aspects of our experimentation.

8.1.2 Stacked Autoencoders in Spatiotemporal Data

We created a system that uses stacked autoencoders to make predictions on spatiotemporal data. This is normally done using convolutional, long short-term memory networks. We did this through careful construction of the design matrices and a data collection method that takes into account local, spatiotemporal influences.

8.1.3 Spatial Generalization

We examined spatial generalization of the stacked autoencoders using the same architecture we used in the previous study. In doing this we showed that adapting trained networks to spatially adjacent areas to perform wind vector determination is reasonable and can yield a useful degree of accuracy. This was purely a generalization experiment, in that the networks did not lose their ability to operate in the original space, as they were not fine-tuned. This provided us with the justification to proceed with exploring ways we could perform transfer learning in the same general domain.

8.1.4 Model Inspired by Numerical Weather Prediction

The model that we created is inspired by the way that numerical weather prediction techniques treat the temporal and spatial components of their data, but uses unsupervised pre-training in feedforward networks rather than the Primitive Equations to make predictions. Regarding NWP, we have some of the same assumptions upon which we arrange and process data. An example of this is that we use a gridding system, which we use to bin the data that we analyzed. We also rely on locality and temporal dependence in the data instances that are run through our procedures.

8.1.5 Obtaining a Baseline for Generalization Performance

In our analysis of the generalization power of our original model, which uses unsupervised pre-training on feedforward networks, we created a method for determining the model's spatial extent of generalization. In determining these baselines we considered the immediate surroundings of an area whose data were used in the training of the network as well as the entire swath of the dataspace.

8.1.6 Dataspace Exploration

We used the mathematical tools of functional data analysis and spatial statistics to explore transferrable encodings of learned characteristics stages of trained an unsupervised pre-trained feedforward network. This was an exploration of the dataspace in terms of it being composed of functional data We used spatial statistics to condition the data and provide a basis for spatial transfer learning. This was the beginning of our transfer learning experimentation.

8.1.7 Controlled Dataspace Analysis

We devloped a highly controlled technique for using autoencoders to analyze a spatiotemporal dataspace. In this technique, we wanted to remove sources of randomness in the training process. This was to emphasize the analysis of the dataset using clones of a single autoencoder so that we could observe how its weights differed when these clones were applied to different areas of the dataspace. We also removed other sources of randomness so the training process for each location would have the same time correspondence. This was so that we could make claims about transferrability of internal encodings that we identified through this process.

8.1.8 Identification of Transferrable Feature Detectors

Using several different techniques, we created ways to extract transferrable feature detectors from pre-trained autoencoders. We cannot claim to know exactly what these feature detectors mean, but we demonstrated their usefulness in providing a starting point for training new models on other areas of the dataspace. This was done using our dual-clustering technique as well as analyzing the way that pre-trained nodes are affected as one moves spatially across the dataspace.

8.1.9 Subset Transfer

Using what we learned in the Dataspace Exploration experiments, we were ready to extend this technique to dividing the dataspace into localized areas for network training. We devised a technique for transferring of subsets of hidden layers (encodings) using locality sensitive clustering of both raw data and layerwise relevance propagation information. We demonstrated the effectiveness of this technique by comparing it to the use of a single global network for making determinations of one of the wind vector components across one day of Hurricane Sandy. We then extended this model to use a nonparametric statistical measure of distributional similarity to determine node transferrability: the Kruskal-Wallis one-way ANOVA test. In doing this, we opened up the technique for other methods of determining transferrability of hidden node subsets.

8.1.10 Locality-Sensitive Clustering Procedure

With the technique for subset-transfer in hand, we created a procedure to increase the area of application of this network through locality-sensitive clustering. This clustering technique takes into account both data clustering and clustering via the feature detectors that were learned during the pre-training procedure. The purpose of this was to optimize the number of neural networks needed to make predictions over a large area.

8.1.11 Hidden Layer Expansion

In order to adapt a neural network to the expanding clusters mentioned in the last section, we had to combine the transferrable feature detectors, creating a network that retained critical characteristics of both networks. To do this we created a way that resolves conflicts between the networks to be combined by expanding the hidden layer to include both of the conflicting nodes. The new architecture was then used to effectively make predictions in the new, expanded cluster.

8.1.12 Cross Dataset Transfer

Having our techniques in hand we applied them from storm to storm on two new datasets: Hurricanes Harvey and Irma. We changed the intent of the prediction in this task from wind vector determination to wind vector prediction. This was to demonstrate the use of our technique in a transductive transfer learning setting, according to the terminology of Pan and Yang [67]. We showed that, in comparison with the best-of-breed configurations from the Subset Transfer experiments, the cross dataset generalization does not provide any discernible benefit, but cross dataset transfer provides a benefit in accuracy. In doing this we showed that there can be an advantage to training on one storm, applying all of the rigor of training on that storm, and fine-tuning on another storm. This reduces the amount of work that needs to be done to obtain useful predictions on the second dataset.

8.1.13 Cluster to Application Mapping

When transferring our clustered models across datasets, the issue of what network from which cluster to apply to each area of interest in the new dataset.

To address this, we created a cluster to application mapping procedure that uses the Kruskal-Wallis one-way ANOVA test to determine which areas in the new dataset are most similar to which clusters in the dataset upon which the model was originally trained. Cluster to application mapping eliminated the need to preprocess the new dataset in the same manner as the original dataset.

8.2 Future Work

The avenues that we have explored have caused many more questions. There are many areas for the continuation of this research, some of which are introduced here.

8.2.1 Application to New Datasets

Our "playground" for the development of our techniques has been meteorological data. These techniques, we believe, are applicable far beyond meteorology or other like Earth sciences. We have mentioned in this dissertation that the data that we are most interested in is spatial, temporal, and functional. Examples of data that is also of this character are in several species of neurological data. Particularly, we have looked at using Quantitative Electroencephalographic (qEEG) [80], Functional Magnetic Resonance Imagery (fMRI) [49], and Function Near Infrared Spectroscopy [78]. All of these appear to be valid avenues of exploration in which we can have a positive impact.

In the future we also do not want to completely depend on space and time as functional predicates for the application of and investigation into these techniques. We recognize that some functional datasets may depend on logical distances rather than spatial or temporal distances. This would be a very interesting area to explore as well.

8.2.2 Extension to Deeper Models

Future generations of this research should be extended to deeper neural network models. Spatial, temporal functional data with any hierarchical expression would benefit from this treatment, as can any type of functional data that has logical dimensions as its functional predicates (as mentioned in the last section). The identification of encodings in this hierarchical internal expression may lead to very interesting ways that the training and application on disparate datasets can exchange a hierarchy of abstract information about their respective areas of concentration...a logical "comparing of notes." Separating what is domain-specific from what is very general and context-preserving may be a very useful consequence of this.

8.2.3 Application to Explainable AI

There is an analogy to be made between subset transfer encodings and mathematical statements, in that they are both encodings of a subset of the big picture. A mathematical statement is a distilled version of some knowledge, sometimes gained from the analysis of data. An encoding that has been refined from an ANN, perhaps in ways related to what we have done here may be the same thing, but in the "natural" terms of ANN's rather than mathematical statements in human terms. Perhaps in the future, work can be done on Neural Network models that exchange these encodings as they would other knowledge encoded in mathematics or word descriptions. Perhaps ontologies and databases of these encodings can be constructed and used so that these bits and pieces of knowledge can be "snapped together" so that a domain-aware model can be the starting point of even newer discoveries.

8.2.4 LSTM and Convolutional Network Application

Having completed this iteration of study using a model that is analogous to how Numerical Weather Prediction is executed, adaption of this logic to the current state of the art in ANN's will be a very interesting endeavor. This shift will be advisable, if not necessary for the treatment of datasets that are spatial and temporal, as these models have demonstrated great success in regression and classification for these types of datasets. Convolutional neural networks have already seen use in meteorology, as have recurrent neural networks like LSTM's, so the trail is actively being blazed in this area. Adaptation will not be without complexities but will be very valuable.

8.2.5 Global Modeling and Temporal Extrapolation

Neural models are very fast and can be adapted to a cellular representation of the Earth. A neural network facsimile of an NWP grid could be created that could be trained against live data. Another generation of this same network could perform extrapolations into the future to perform worldwide weather predictions. Given the speed at which these models work, this could become a competitor to NWP models currently existing. This model could allow the entire atmosphere to be tessellated, where favorable conditions for, say, air travel could be determined based on parameters that are fed to the system. Creating a database of encodings and trained networks for use on future storms would be an ambitious, but not impossible goal, and would help in the global modeling effort.

8.3 Final Remarks

We began this study with a question in our minds about enhancing radiometric datasets so they could be leveraged to provide more information than what they were designed to provide, namely that of wind vectors. Alone, this will be useful, considering all of the datasets it can enhance and all of the information it could provide. But there was always an inkling of greater utility lurking in the background, which could be seen in greater relief once we leveraged the tools of spatial statistics, functional data analysis, and the rest of computer science.

Neural networks have seen an explosive increase in their use in a diversity of fields. The identification of transferrable encodings in autoencoders is a significant advancement which can be used in any of them to foster even faster development and acceptance of the models. This is because, in every field of application, there is a great variety of knowledge that has to be assimilated by each model. If all of this knowledge could be encoded, catalogued, and exchanged with other models, new models would have more intelligent points from which to start their training and application. This would lead to greater accuracy, shorter training times, and even newer encodings discovered and shared, or marketed, with the rest of the universe of neural networks.

In conclusion, we look forward to continuing this work, adapting it to the state of the art in the rest of the neural network community. We are optimistic about being able to refine what we have done here, making the boundary between transferrable encodings and noise much clearer. And we are excited to find diverse uses for these techniques as they become more developed.

REFERENCES CITED

[1] DARPA XAI BAA. `https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf`, 2016.

[2] The Weather Company Launches 'Deep Thunder' — The Weather Company. `http://www.theweathercompany.com/DeepThunder`, 2016.

[3] Hurricane Irma Approaches Puerto Rico — NOAA National Environmental Satellite, Data, and Information Service (NESDIS), 2020.

[4] MicroMAS 1. `https://directory.eoportal.org/web/eoportal/satellite-missions/m/micromas-1`, 2020.

[5] NOAA Satellite Imagery of Hurricane Harvey — NOAA National Environmental Satellite, Data, and Information Service (NESDIS), 2020.

[6] C. Abbe. The Physical Basis of Long Range Weather Forecasts. *Monthly Weather Review*, 29(12):551–561, 12 1901.

[7] K. Abhishek, A. Kumar, R. Ranjan, and S. Kumar. A Rainfall Prediction Model Using Artificial Neural Network. *IEEE Control and System Graduate Research Colloquium*, 7:82–87, 2012.

[8] K. Abhishek, M. Singh, S. Ghosh, and A. Anand. Weather Forecasting Model Using Artificial Neural Network. *Procedia Technology*, 4:311–318, 2012.

[9] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A Learning Algorithm for Boltzmann Machines*. *Cognitive Science*, 9:147–169, 1985.

[10] C. Ambroise, A. Dehman, P. Neuvial, G. Rigaill, and N. Vialaneix. Adjacency-constrained hierarchical clustering of a band similarity matrix with application to genomics. *Algorithms for Molecular Biology*, 14(1):22, 11 2019.

[11] American Meteorological Society. American Meteorological Society Glossary of Meteorology, 2011.

[12] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140, 7 2015.

[13] M. Bachmaier and M. Backes. Variogram or Semivariogram? Variance or Semivariance? Allan Variance or Introducing a New Term? *Mathematical Geosciences*, 43(6):735–740, 8 2011.

[14] P. Baldi. Autoencoders, Unsupervised Learning, and Deep Architectures. *JMLR Workshop on Unsupervised and Transfer Learning*, 27:37–50, 2012.

[15] R. Barnes. dggridR: Discrete Global Grids for R. `https://cran.r-project.org/web/packages/dggridR/vignettes/dggridR.html`, 2016.

[16] L. Bel, A. Bar-Hen, R. Petit, and R. Cheddadi. Spatio-temporal functional regression on paleoecological data. *Journal of Applied Statistics*, 38(4):695–704, 4 2011.

[17] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy Layer-Wise Training of Deep Networks. *Advances in Neural Information Processing Systems*, pages 153–160, 2007.

[18] A. Binder, S. Bach, G. Montavon, K. R. Müller, and W. Samek. Layer-wise relevance propagation for deep neural network architectures. In *Lecture Notes in Electrical Engineering*, volume 376, pages 913–922, 2016.

[19] V. Bjerknes. The problem of weather forecasting as a problem in mechanics and physics. *Meteorologische Zeitschrift*, 21:1–7, 1904.

[20] Brian Jackson. IBM Partners with The Weather Company to Update Forecasts 26 Billion Times a Day. `https://www.itbusiness.ca`, 2015.

[21] B. Cao, S. J. Pan, Y. Zhang, D. Y. Yeung, and Q. Yang. Adaptive transfer learning. *Proceedings of the National Conference on Artificial Intelligence*, 1:407–412, 2010.

[22] A. S. Cofino, R. Cano, C. Sordo, and J. M. Gutierrez. Bayesian Networks for Probabilistic Weather Prediction . *Proceedings of the 15th European Conference on Artificial Intelligence*, 700:695–700, 2002.

[23] S. N. Collins, R. S. James, P. Ray, K. Chen, A. Lassman, and J. Brownlee. Grids in Numerical Weather and Climate Models. *Climate Change and Regional/Local Responses*, page 256, 5 2013.

[24] M. Dalto, J. Matusko, and M. Vasak. Deep neural networks for ultra-short-term wind forecasting. In *Proceedings of the IEEE International Conference on Industrial Technology*, volume 2015-June, pages 1657–1663, 2015.

[25] M. Dixon, G. Wiener, M. Dixon, and G. Wiener. TITAN: Thunderstorm Identification, Tracking, Analysis, and Nowcasting—A Radar-based Methodology. *Journal of Atmospheric and Oceanic Technology*, 10(6):785–797, 1993.

[26] D. Erhan, Y. Bengio, A. Courville, P. Vincent, and S. Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

[27] A. Fallis. Tropical Cyclone Report Hurricane Sandy (AL182012) 22 – 29 October 2012 Eric. *Journal of Chemical Information and Modeling*, 53(9):1689–1699, 2013.

[28] Q. Y. Feng, R. Vasile, M. Segond, A. Gozolchiani, Y. Wang, M. Abel, S. Havlin, A. Bunde, and H. A. Dijkstra. ClimateLearn: A machine-learning approach for climate prediction using network measures. *Geoscientific Model Development Disucssions*, (February):1–18, 2016.

[29] R. J. Firth. *A Novel Recurrent Convolutional Neural Network for Ocean and Weather Forecasting.* PhD thesis, Louisiana State University, 2016.

[30] D. H. Fisher. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning*, 2(2):139–172, 1987.

[31] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[32] K. Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20(3-4):121–136, 9 1975.

[33] A. Gasiewski. Personal Communication with Director of the Center for Environmental Technology, University of Colorado, 2017.

[34] A. Gasiewski, D. Jackson, J. Wang, P. Racette, and D. Zacharias. Airborne imaging of tropospheric emission at millimeter and submillimeter wavelengths. In *Proceedings of IGARSS '94 - 1994 IEEE International Geoscience and Remote Sensing Symposium*, volume 2, pages 663–665. IEEE.

[35] A. E. Gelfand, P. J. Diggle, M. Fuentes, and P. Guttorpt. *Handbook of Spatial Statistics*, volume 20103158. 2010.

[36] R. Gibb, A. Raichev, and M. Speth. The Rhealpix Discrete Global Grid System. `http://www.discreteglobalgrids.org/wp-content/uploads/sites/33/2016/10/PlanetRiskDGGS.pdf`.

[37] E. Ginzburg, T. L. Zakrison, G. D. Pust, A. A. Grant, N. Lu, R. Rattan, and N. Namias. Hurricane Irma. *National Hurricane Center*, 85(3):635–636, 2018.

[38] A. D. E. Giorgio and A. Holst. *A study on the similarities of Deep Belief Networks and Stacked A study on the similarities of Deep Belief Networks and Stacked Autoencoders Supervisor.* PhD thesis, KTH Royal Institute of Technology, 2015.

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

[40] A. Grover and E. Horvitz. A Deep Hybrid Model for Weather Forecasting. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 379–386, 2015.

[41] S. S. Haykin. *Neural networks and learning machines.* Prentice Hall/Pearson, 2009.

[42] E. Hernández, V. Sanchez-Anguix, V. Julian, J. Palanca, and N. Duque. Rainfall prediction: A deep learning approach. In *Hybrid Artificial Intelligent Systems (HAIS)*, pages 151–162, 2016.

[43] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets *. *Neural Computation*, 18(7):1527–1554, 2006.

[44] Q. Hu, R. Zhang, and Y. Zhou. Transfer learning for short-term wind speed prediction with deep neural networks. *Renewable Energy*, 85:83–95, 1 2016.

[45] IBM. Deep Thunder. `https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepthunder/`, 2020.

[46] Jennifer Booton. IBM Finally Reveals Why It Bought The Weather Company. `https://www.marketwatch.com/story/ibm-finally-reveals-why-it-bought-the-weather-company-2016-06-15`, 2016.

[47] H. E. Jones and N. Bayley. The Berkeley Growth Study. *Child Development*, 12(2):167–173, 6 1941.

[48] H. Kamyshanska and R. Memisevic. The Potential Energy of an Autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014.

[49] M. Khosla, K. Jamison, G. H. Ngo, A. Kuceyeski, and M. R. Sabuncu. Machine learning in resting-state fMRI analysis. *Magnetic Resonance Imaging*, 2019.

[50] K. King. *Functional Data Analysis With Application to United States Weather Data.* PhD thesis, 2014.

[51] W. H. Kruskal and W. A. Wallis. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260):583–621, 12 1952.

[52] J. K.Williams, D.A.Ahijevych, C.J.Kessinger, T.R.Saxen, M.Steiner, and S.Dettling. A Machine Learning Approach to Finding Weather Regimes and Skillful Predictor Combinations for Short-Term Storm Forecasting. *National Center for Atmospheric Research*, pages 1–6, 2008.

[53] H. Li, G. A. Perez, C. A. Martinez, and A. E. Mynett. Self-learning cellular automata for forecasting precipitation from radar images. *Journal of Hydrologic Engineering*, 18(2):206–211, 2 2013.

[54] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu. Understanding of internal clustering validation measures. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 911–916, 2010.

[55] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang. Transfer Learning using Computational Intelligence: A Survey. *Knowledge Based Systems*, 80(10):14–23, 2015.

[56] P. Lynch. The origins of computer weather prediction and climate modeling. *Journal of Computational Physics*, 227(7):3431–3444, 2008.

[57] J. Mason. Numerical Weather Prediction. *P. Roy. Soc. Lond. A Mat.*, 407:51–60, 1986.

[58] R. McAllister and J. Sheppard. Evaluating Spatial Generalization of Stacked Autoencoders in Wind Vector Determination. In *FLAIRS Conference*, Melbourne, FL, 2018.

[59] R. McAllister and J. Sheppard. Exploring Transferability in Deep Neural Networks with Functional Data Analysis and Spatial Statistics. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 2019-July, 2019.

[60] R. A. McAllister and J. W. Sheppard. Deep Learning for Wind Vector Determination. In *IEEE Symposium Series on Computational Intelligence*, Honolulu, HI, 2017.

[61] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 12 1943.

[62] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 65:211–222, 5 2017.

[63] S. Narejo and E. Pasero. Meteonowcasting using Deep Learning Architecture. *IJACSA) International Journal of Advanced Computer Science and Applications*, 8(8):16–23, 2017.

[64] NASA. Remote Sensors. `https://earthdata.nasa.gov/user-resources/remote-sensors`, 2015.

[65] M. A. Oliver and R. Webster. *Basic Steps in Geostatistics: The Variogram and Kriging*. Springer Briefs in Agriculture. Springer International Publishing, Cham, 2015.

[66] T. L. Paine, P. Khorrami, W. Han, and T. S. Huang. An Analysis of Unsupervised Pre-training in Light of Recent Advances. In *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015.

[67] S. J. Pan and Q. Yang. a Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 1(10):1345–1359, 2010.

[68] H. A. Panofsky. The Planetary Boundary Layer. *Advances in Geophysics*, 28(PB):359–385, 1985.

[69] G. Percivall. Geodata fusion study by the Open Geospatial Consortium. `http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2016226`, 5 2013.

[70] A. Perez-Vega, C. M. Travieso, J. G. Hernandez-Travieso, J. B. Alonso, M. K. Dutta, and A. Singh. Forecast of temperature using support vector machines. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 388–392. IEEE, 2016.

[71] A. S. Qureshi and A. Khan. Adaptive transfer learning in deep neural networks: Wind power prediction using knowledge transfer from region to region and between different task domains. *Computational Intelligence*, 35(4):1089–1113, 2019.

[72] A. S. Qureshi, A. Khan, A. Zameer, and A. Usman. Wind power prediction using deep neural network based meta regression and transfer learning. *Applied Soft Computing*, 58:742–755, 9 2017.

[73] Y. Radhika and M. Shashi. Atmospheric Temperature Prediction using Support Vector Machines. *International Journal of Computer Theory and Engineering*, 1(1):55–58, 2009.

[74] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein. On the expressive power of deep neural networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 6, pages 4351–4374, 6 2017.

[75] A. Rakhmatova, A. Sergeev, A. Buevich, A. Shichkin, and M. Sergeeva. Partition Procedure of the Initial Data for the Models Based on Artificial Neural Networks. In *Proceedings - 2019 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology, USBEREIT 2019*, pages 241–243. Institute of Electrical and Electronics Engineers Inc., 4 2019.

[76] M. A. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems*, pages 1137–1144, 2007.

[77] T. Rao, N. Rajasekhar, and T. V. Rajinikanth. An efficient approach for Weather forecasting using Support Vector Machines. In *2012 International Conference on Intelligent Network and Computing (ICINC 2012)*, 2012.

[78] J. Rhee and R. K. Mehta. Functional connectivity during handgrip motor fatigue in older adults is obesity and sex-specific. *Frontiers in Human Neuroscience*, 12, 11 2018.

[79] A. D. Ross. Hurricane Harvey. In *U.S. Emergency Management in the 21st Century*, pages 91–122. 2020.

[80] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert. Deep learning-based electroencephalography analysis: a systematic review Deep learning-based electroencephalography analysis: a systematic review Topical Review. *J. Neural Eng*, 16:37, 2019.

[81] D. Rumelhart, G. E. Hinton, and R. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1: Foundat. MIT Press, Cambridge, MA, 1986.

[82] K. Sahr, D. White, and A. J. Kimerling. Geodesic Discrete Global Grid Systems Discrete Global Grid Systems: Basic Definitions. *Cartography and Geographic Information Science*, 30(2):121–134, 2003.

[83] A. G. Salman, B. Kanigoro, and Y. Heryadi. Weather forecasting using deep learning techniques. *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 281–285, 2015.

[84] W. Samek, G. Montavon, A. Binder, S. Lapuschkin, and K.-R. Müller. Interpreting the Predictions of Complex ML Models by Layer-wise Relevance Propagation. In *Workshop on Interpretable Machine Learning for Complex Systems (NEURIPS 2016)*, Barcelona, Spain, 2016.

[85] W. Samek, T. Wiegand, and K.-R. Muller. Explainable Artificial Intelligence: Understanding, Visualizing, and Interpreting Deep Learning Models. *ITU Journal: ICT Discoveries*, (Special Issue No. 1), 2017.

[86] V. Satopää, J. Albrecht, D. Irwin, and B. Raghavan. Finding a ”kneedle” in a haystack: Detecting knee points in system behavior. In *Proceedings - International Conference on Distributed Computing Systems*, pages 166–171, 2011.

[87] M. Schwartz, J. Barrett, P. Fieguth, P. Rosenkranz, M. Spina, and D. Staelin. Passive microwave imagery of a tropical storm near 118 GHz thermal and precipitation structure. In *Proceedings of IGARSS '94 - 1994 IEEE International Geoscience and Remote Sensing Symposium*, volume 4, pages 2433–2435.

[88] A. P. Sergeev, A. G. Buevich, E. M. Baglaeva, and A. V. Shichkin. Combining spatial autocorrelation with machine learning increases prediction accuracy of soil heavy metals. *Catena*, 174:425–435, 3 2019.

[89] B. W. Silverman and J. O. Ramsay. *Functional Data Analysis*. Springer, 2005.

[90] S. Singh and J. Gill. Temporal Weather Prediction using Back Propagation based Genetic Algorithm Technique. *International Journal Intelligent Systems and Applications*, 12(November):55–61, 2014.

[91] V. Singh. Application of Artificial Neural Networks for Predicting Generated Wind Power. *International Journal of Advanced Computer Science and Applications(ijacsa)*, 7(3):250–253, 2016.

[92] S. Siu, S.-S. Yang, C.-M. Lee, and C.-L. Ho. Improving the Back-Propagation Algorithm Using Evolutionary Strategy. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 54(2):171–175, 2 2007.

[93] R. V. Soelen and J. W. Sheppard. Pruned Networks for Transfer Learning. In *IEEE International Joint Conference on Neural Networks*, 2019.

[94] R. Stull. *Practical Meteorology - An Algebra-based Survey of Atmospheric Science*. Pacific Grove: Brooks Cole, 2012.

[95] A. C. Subhajini. Application of Neural Networks in Weather Forecasting. *Climate Change and Conservation Research*, 4(1):8–18, 2018.

[96] E. Suli, E. Suli, D. Mayers, and D. Mayers. *An Introduction to Numerical Analysis*. Wiley, 2003.

[97] Suomi-NPP. Suomi National Polar-orbiting Partnership. `https://www.nasa.gov/mission_pages/NPP/main/index.html`, 2017.

[98] L. Torrey and J. Shavlik. Transfer Learning. *Machine Learning*, pages 1–22, 2009.

[99] Y. Trokhimovski and V. Irisov. The analysis of wind exponents retrieved from microwave radar and radiometric measurements. *IEEE Transactions on Geoscience and Remote Sensing*, 38(1):470–479, 2000.

[100] J. Wang, P. Balaprakash, and R. Kotamarthi. Fast domain-aware neural network emulation of a planetary boundary layer parameterization in a numerical weather forecast model. *Geoscientific Model Development*, 12(10):4261–4274, 10 2019.

[101] T. T. Warner. *Numerical Weather and Climate Prediction.* Cambridge University Press, 2011.

[102] C.-C. Wei. Wavelet Support Vector Machines for Forecasting Precipitation in Tropical Cyclones: Comparisons with GSVM, Regression, and MM5. *Weather and Forecasting*, 27(2):438–450, 4 2012.

[103] C.-C. Wei. Study on Wind Simulations Using Deep Learning Techniques during Typhoons: A Case Study of Northern Taiwan. *Atmosphere*, 10(11):684, 11 2019.

[104] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 12 2016.

[105] E. W. Weisstein. Spline. `https://mathworld.wolfram.com/Spline.html`.

[106] J. R. Wertz, D. F. Everett, and J. J. Puschell. *Space mission engineering : the new SMAD.* Microcosm Press, 2011.

[107] A. Wimmers, C. Velden, and J. H. Cossuth. Using Deep Learning to Estimate Tropical Cyclone Intensity from Satellite Passive Microwave Imagery. *Monthly Weather Review*, 147(6):2261–2282, 6 2019.

[108] Q. Yang, Y. Zhang, W. Dai, and S. J. Pan. *Transfer Learning.* Cambridge University Press, 1 2020.

[109] K. Zhang and A. J. Gasiewski. Microwave CubeSat fleet simulation for hydrometric tracking in severe weather. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 5569–5572, 2016.

[110] Y. Zhang, J. Hemperly, N. Meskhidze, and W. C. Skamarock. The Global Weather Research and Forecasting (GWRF) Model: Model Evaluation, Sensitivity Study, and Future Year Simulation. *Atmospheric and Climate Sciences*, 2(July):231–253, 2012.