A Novel Recurrent Convolutional Neural Network for Ocean and Weather Forecasting

Robert James Firth
Louisiana State University and Agricultural and Mechanical College, robertfirth@live.com

A NOVEL RECURRENT CONVOLUTIONAL NEURAL NETWORK FOR OCEAN AND WEATHER FORECASTING

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical
College in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science and Electrical Engineering

by
Robert James Firth
B.S., Louisiana State University, 2010
May 2016

ACKNOWLEDGEMENTS

I am very grateful to Dr. Jianhua Chen for her assistance, guidance, and many helpful discussions over the course of the last several years.

I am very grateful to Dr. Konstantin Busch and Dr. Jian Zhang for volunteering to serve on my committee, along with Dr. Mehdi Farasat for serving as the dean's representative.

I am grateful to Dr. Kevin Robbins from the NOAA Southern Regional Climate Center at LSU and Dr. Frederick Carr from the University of Oklahoma for consultations and discussions on meteorology and climate models.

I am very thankful for the guidance of Dr. Nan Walker from the LSU Earth Scan Laboratory for teaching me so much about our oceans and remote sensing.

And I am very appreciative of the information and insights into the RAP model provided by Ben Kyger, Luis Cano, Rebecca Cosgrove, Scott Cross, and Simon Hsiao at the National Centers for Environmental Prediction, and the information about the Navy NCOM model provided by Allan Wallcraft, Joe Metzger, Charlie Barron, and Dong Ko from the Navy Research Lab at the Stennis Space Center.

Table of Contents

Acknowledgements ii

List of Tables v

List of Figures vii

Abstract x

1 Introduction 1
- 1.1 Motivations 1
- 1.2 Problem Statement and Objectives 2

2 Literature Review 3
- 2.1 Neural networks 3
- 2.2 Noisy Data, Data Assimilation, and Kalman Filter 10
- 2.3 Choice of Neural Network or Genetic Algorithm for Learning 10
- 2.4 Numerical Weather Prediction 11
- 2.5 Parallelization and Scalability 14
- 2.6 Summary 16

3 Methodology 17
- 3.1 Introduction 17
- 3.2 Differentiation-Integration Time Step Network 17
- 3.3 Recurrent Convolutional DITS Network 18
- 3.4 Parallelization 21

4 Case Study: Ocean Modelling 24
- 4.1 Introduction 24
- 4.2 Motivation 24
- 4.3 LOM: Learned Ocean Model 25
- 4.4 Analysis 27
- 4.5 Results 27
- 4.6 Pruning 33
- 4.7 Discussion 33

5 Case Study: Atmospheric Modeling 35
- 5.1 Introduction 35
- 5.2 LM3: Learned Mesoscale Meteorological Model 36
- 5.3 HLM3: Hurricane Learned Mesoscale Meteorological Model 37
- 5.4 Ensemble Forecasting 38

5.5 Implementation and experimental setup 38
5.6 Data 38
5.7 Results 39
5.8 Pruning 46
5.9 Training Error vs Epoch 48
5.10 Discussion 48

6 Results of Parallelization 51
6.1 Introduction 51
6.2 Results 51

7 Conclusion 53
7.1 Summary 53
7.2 Benefits of the Proposed Approach 53

8 Further Study 55

References 56

Appendix 61

Vita 63

List of Tables

Table 2-1: List of variables and their meaning. 12

Table 4-1 Summary of results forecasting salinity for 8 levels of the ocean, where LOM represents the results of the proposed method. Error is MAE. 29

Table 4-2 Table of results for levels where LOM performed better than NCOM, persistence, and climatology. 29

Table 4-3 Summary of results forecasting temperature for 8 levels of the ocean, where LOM represents the results of the proposed method. Error is MAE in Celsius. 30

Table 4-4 Table of results for temperature forecasting for levels where LOM performed better than NCOM. Error is MAE. 31

Table 4-5 Summary of results forecasting temperature for 8 levels of the ocean, where LOM represents the results of the proposed method. Error is MAE in Celsius. 32

Table 4-6 Summary of results forecasting temperature for 8 levels of the ocean, where LOM represents the results of the proposed method. Error is MAE in Celsius. 32

Table 5-1 Summary of results forecasting U for 10 levels of the atmosphere, where LM3 represents the results of the proposed method. Error is MAE in m/s (meters per second) 41

Table 5-2 Layers where LM3 is more accurate than RAP at forecasting the u-component of the wind 41

Table 5-3 Summary of results forecasting V for 10 levels of the atmosphere, where LM3 represents the results of the proposed method. Error is MAE in m/s (meters per second) 42

Table 5-4 Summary of results forecasting temperature for 10 levels of the atmosphere, where LM3 represents the results of the proposed method. Error is MAE in Celsius. 44

Table 5-5 Layers where LM3 is more accurate than RAP for forecasting temperature 44

Table 5-6 Summary of results for forecasting relative humidity for levels that are multiples of 100 45

Table 5-7 Layers where LM3 is more accurate than RAP for forecasting relative humidity 45

Table 5-8 Summary of results for forecasting specific humidity for levels that are multiples of 100 46

Table 5-9 Layers where LM3 is more accurate than RAP for forecasting specific humidity 46

Table 5-10 Table of variables and layers, and the network that best forecasts that variable. 47

LIST OF FIGURES

FIGURE 1-0-1: A TORNADO OUTSIDE DAMMIT, TX ON JUNE 2, 1995 (HARALD RICHTER). HURRICANE MITCH ON OCTOBER 26, 1998 (LSU EARTH SCAN LAB). FLOODING CAUSED BY HURRICANE WILMA ON OCTOBER 24, 2005 (MARC AVERETTE). SNOW IN BOSTON, MA ON JANUARY 28, 2015 (AP/ELISE AMENDOLA) 1

FIGURE 1-0-2: SURFACE WATER CURRENT VELOCITY VECTOR FIELD OF THE GULF OF MEXICO. 2

FIGURE 2-1: AN ARTIFICIAL NEURON COMPUTES THE WEIGHTED SUM OF THE INPUTS, THEN COMPUTES THE OUTPUT USING AN ACTIVATION FUNCTION. 3

FIGURE 2-2: A RECURRENT NEURAL NETWORK WHERE THE OUTPUT OF THE HIDDEN NEURON IS FED BACK AS AN INPUT TO THE SAME NEURON. 5

FIGURE 2-3: ON THE LEFT, A RECURRENT NETWORK UNROLLED. THIS REPRESENTS FORWARD PROPAGATION THROUGH TIME. ON THE RIGHT, THE BACKWARDS PROPAGATION OF ERROR THROUGH THE SAME NETWORK. 5

FIGURE 2-4: AN AUTOENCODER NETWORK. 6

FIGURE 2-5: A CONVOLUTIONAL NEURAL NETWORK, SIMILAR TO LENET-5 [10], USED FOR OBJECT RECOGNITION IN IMAGES. 7

FIGURE 2-6: AN EXAMPLE OF A CONVOLUTION FOLLOWED BY A MAXPOOLING OPERATION. 7

FIGURE 2-7: AN EXAMPLE OF A 2X2 MAXPOOLING COMPUTATION. THE IMAGE IS SPLIT INTO 2X2 REGIONS, AND THE MAXIMUM VALUE FROM EACH REGION IS SAVED. THIS RESULTS IN AN OUTPUT IMAGE THAT IS ¼ THE SIZE OF THE ORIGINAL. 8

FIGURE 2-8: ABOVE IS A 4X4 GRID. A SINGLE PROCESSOR COULD COMPUTE THE FORECAST FOR ALL 16 LOCATIONS, OR 4 PROCESSORS COULD BE USED IN PARALLEL TO EACH COMPUTE ONE OF THE FOUR QUADRANTS. THIS WOULD LEAD TO A 75% REDUCTION IN COMPUTATION TIME. 15

FIGURE 3-1: A FORWARD INTEGRATION DITS NETWORK 17

FIGURE 3-2: A CENTERED-IN-TIME DITS NETWORK 17

FIGURE 3-3: A RECURRENT CONVOLUTIONAL DITS NETWORK. ON THE LEFT, EACH PLANE REPRESENTS A 2D INPUT VARIABLE, SUCH AS TEMPERATURE, HUMIDITY, OR GEOPOTENTIAL height. In the center we have the derivatives of each of the input variables with respect to time. At the end, we have the output variables one time-step in the future. 19

Figure 3-4: Two applications of the Recurrent Convolutional DITS Network from Figure 3-3 unrolled to show a forecast two time steps into the future. 19

Figure 3-5: Three applications of a DITS network with a varying lambda. 20

Figure 3-6 Spatial decomposition for parallelization 21

Figure 3-7: Forward propagation followed by backward propagation through a network with three layers, or a network recurrently applied three times 22

Figure 3-8: Pipelined forward and backward propagation across three machines. 23

Figure 4-1: Surface water current velocity vector field of the Gulf of Mexico. The Loop Current is the dominating feature in the Eastern half of the Gulf. Data provided by CCAR and Dr. Robert Leben. 24

Figure 4-2: On the left, a typical water current image of the Loop Current in the Gulf of Mexico. On the right, a very erratic Loop Current extends significantly further north. Data provided by CCAR and Dr. Robert Leben. 25

Figure 4-3: NCOM input data depicting water current velocity for the Caribbean Sea and Gulf of Mexico. 26

Figure 4-4 Time-step Size vs Execution Time for LOM and NCOM for a 1 hour forecast 28

Figure 4-5 LOM salinity forecast at the surface. On the left, a comparison of the forecast change in salinity vs actual change in salinity. On the right, actual salinity vs forecasted salinity. 30

Figure 4-6 LOM Water temperature forecast. On the left, scatterplot of forecasted change in temperature vs actual change in temperature at the surface. 31

Figure 4-7 LOM water current forecast. On the left, scatterplot of actual change in surface water current for the u (east-west) component vs forecasted change. On the right, the same, except for the v (north-south) component. 33

Figure 5-1: Networks N1 through N6 are independent and can be trained separately. Each network computes the time derivative of a different meteorological observable variable. 35

Figure 5-2 Temperature forecasting recurrent network unfolded though time with time step Δt equal to 30 minutes. U and V are the east-west and north-south components of the wind speed, respectively. T is the temperature, C is the cloud cover, and S is the sun's altitude above the horizon. Outputs of the network do not directly re-enter as inputs. Instead, they must go through a pre-processing stage, which complicates backpropagation training. 36

Figure 5-3: NCEP Grid 252, a 301x225x37 grid of observations with approximately 20km horizontal resolution. Only the 56x49 grid over the southeastern US is used. This represents a roughly homogenous region with similar elevation and no ocean 36

Figure 5-4: My 10' NOAAPORT dish. Due to large download sizes, going forward input data will be captured from the NOAAPORT satellite broadcast network. 39

Figure 5-5 Time-step Size vs Execution Time for LM3 and RAP for a 1 hour forecast 40

Figure 5-6: Scatterplot for U Forecast on 1000mb Level 42

Figure 5-7: Scatterplot for U Forecast on 500mb Level 43

Figure 5-8: Scatterplot for U Forecast on 100mb Level 43

Figure 5-9 Training error vs epoch. 48

Abstract

Numerical weather prediction is a computationally expensive task that requires not only the numerical solution to a complex set of non-linear partial differential equations, but also the creation of a parameterization scheme to estimate sub-grid scale phenomenon.

The proposed method is an alternative approach to developing a mesoscale meteorological model – a modified recurrent convolutional neural network that learns to simulate the solution to these equations.

Along with an appropriate time integration scheme and learning algorithm, this method can be used to create multi-day forecasts for a large region. The learning method presented is an extended form of Backpropagation Through Time for a recurrent network with outputs that feed back through as inputs only after undergoing a fixed transformation.

An initial implementation of this approach has been created that forecasts for 2,744 locations across the southeastern United States at 36 vertical levels of the atmosphere, and 119,000 locations across the Atlantic Ocean at 39 vertical levels. These models, called LM3 and LOM, forecast wind speed, temperature, geopotential height, and rainfall for weather forecasting and water current speed, temperature, and salinity for ocean forecasting.

Experimental results show that the new approach is 3.6 times more efficient at forecasting the ocean and 16 times more efficient at forecasting the atmosphere.

The new approach showed forecast skill by beating the accuracy of two models, persistence and climatology, and was more accurate than the Navy NCOM model on 16 of the first 17 layers of the ocean below the surface (2 meters to 70 meters) for forecasting salinity and 15 of the first 17 layers for forecasting temperature. The new approach was also more accurate than the RAP model at forecasting wind speed on 7 layers, specific humidity on 7 layers, relative humidity on 6 layers, and temperature on 3 layers, with competitive results elsewhere.

1 Introduction

1.1 Motivations

Every hour, a global network of sensors produces a mountain of data about the current state of our oceans and atmosphere. This data joins vast archives going back over a hundred years.

Weather forecasts are important to the energy industry for forecasting electrical power demand, to agriculture for crop planting and harvest, event planning, and many other industries. Indeed, our everyday lives are impacted by the weather.

Figure 1-0-1: A tornado outside Dammit, TX on June 2, 1995 (Harald Richter). Hurricane Mitch on October 26, 1998 (LSU Earth Scan Lab). Flooding caused by Hurricane Wilma on October 24, 2005 (Marc Averette). Snow in Boston, MA on January 28, 2015 (AP/Elise Amendola).

Ocean current forecasts are essential to offshore oil production [1]. Strong ocean currents in coastal Louisiana caused by the periodic northern migration of the Loop Current caused significant disruption and downtime as recently as late 2014 [2, 3].

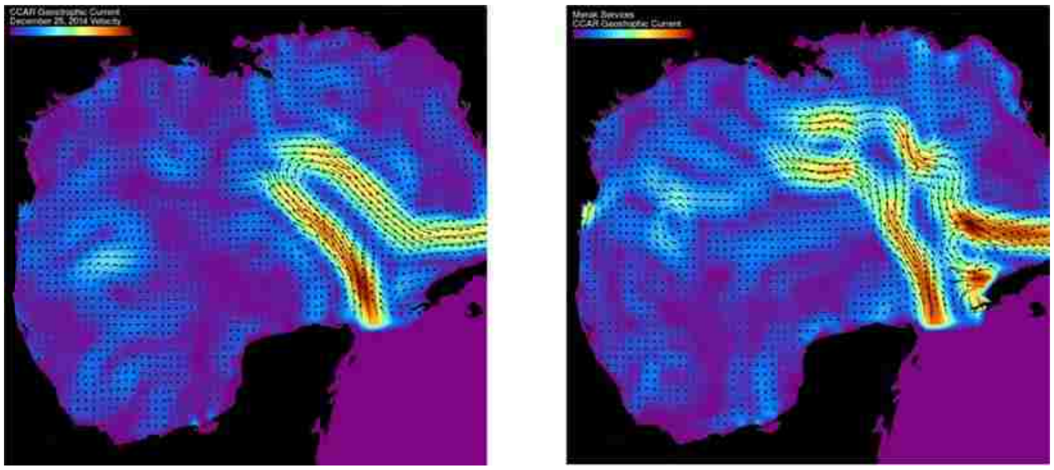


Figure 1-0-2: Surface water current velocity vector field of the Gulf of Mexico.

1.2 Problem Statement and Objectives

Given a horizontal $m \times n$ grid of v observations on h vertical layers at time t and (optionally) $t - \Delta t$, calculate forecasted future values of those v observations at times $t + s\Delta t$, where s is a positive integer greater than 0 and Δt is a time step.

The objective of the present work is to propose a novel method for forecasting the ocean and atmosphere using a machine learning approach. The proposed method is used to implement a full mesoscale meteorological model that creates forecasts for a 56 x 49 x 37 region centered over the southeastern United States and an ocean model that creates forecasts for a 350 x 340 x 39 region of the Atlantic Ocean.

The learning task is to learn to compute the partial derivative of each variable with respect to time. This is coupled with a centered-in-time time integration technique to time step every variable forward.

The recurrent portion of the network refers to its repeated application until a forecast of the desired length is produced.

This is accomplished by using a model similar to a convolutional neural network. The same learned weights are used for all spatial positions in a given layer of the model. This results in a significant reduction in the number of weights to learn, compared to a fully connected network.

We want to create an efficient method that is scalable and produces competitive prediction accuracy. This is accomplished by carefully designing algorithms that are easily parallelized.

2 Literature Review

2.1 Neural networks

Artificial neural networks, developed by McCulloch and Pitts in 1943, are loosely modelled after the biological neural networks in our brains [4]. They consist of a network of neurons connected by weights. The output of a neuron is the weighted sum of the inputs, followed by an activation function.

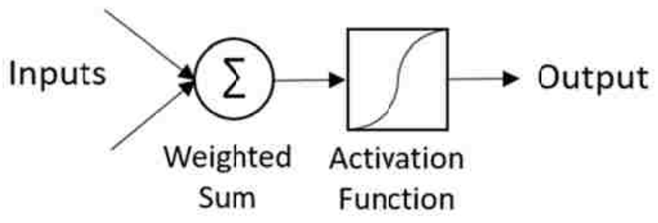
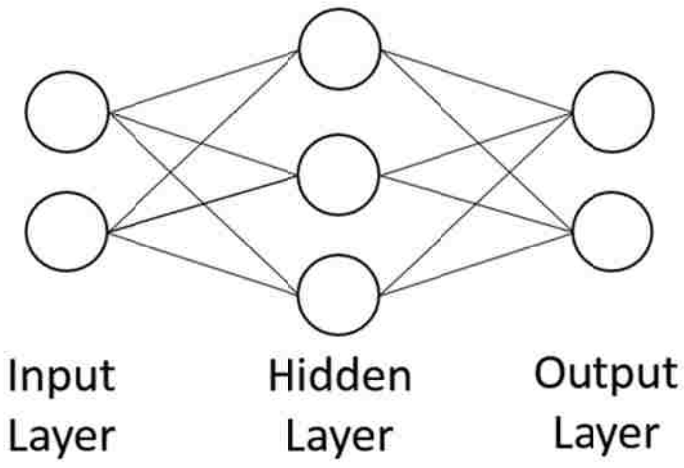


Figure 2-1: An artificial neuron computes the weighted sum of the inputs, then computes the output using an activation function.

These neurons can be connected into large multilayer networks. These networks are useful because they are universal approximators – given sufficient size and training, they are able to simulate any function [5].



Training large multilayer networks became possible with the development of the backpropagation algorithm [6]. Training deep networks can be preceded by unsupervised learning to speed up subsequent supervised learning [7].

2.1.1 Backpropagation of Error

The Backpropagation Algorithm is essential to training neural networks. It computes how the values of the weights should change to minimize the error using gradient descent.

We compute error at the output as [5]:

$$\text{Error} = \frac{1}{2}\sum_{k=0}^{n-1}(t_k - y_k)^2 \tag{2-1}$$

Where t_k is the kth target value, y_k is the kth output value, and n is the number of outputs.

We can compute the error at the output as [6]:

$$\delta_{ok} = (t_k - y_k)y_k(1 - y_k) \tag{2-2}$$

And the weight update for the output layer [6]:

$$w_{jk} = w_{jk} + \eta\delta_{ok}a_j^{hidden} \tag{2-3}$$

Where w_{jk} is the weight between the jth hidden neuron and the kth output neuron, a_j^{hidden} is the activation of the jth hidden neuron, and η is the learning rate.

Additionally, we can compute the error at the hidden layer as [6]:

$$\delta_{hj} = a_j(1 - a_j)\sum\nolimits_k w_{jk}\delta_{ok} \tag{2-4}$$

And the weight update for the hidden layer [6]:

$$w_{ij} = w_{ij} + \eta\delta_{hj}x_i \tag{2-5}$$

Where w_{ij} is the weight between the ith input node and jth node of the hidden layer, and x_i is the ith input.

2.1.2 Recurrent Neural Networks

A recurrent neural network simply means that the network has outputs that are fed back as inputs. These outputs may either be the output of the entire network, or they could be the output of a hidden layer.

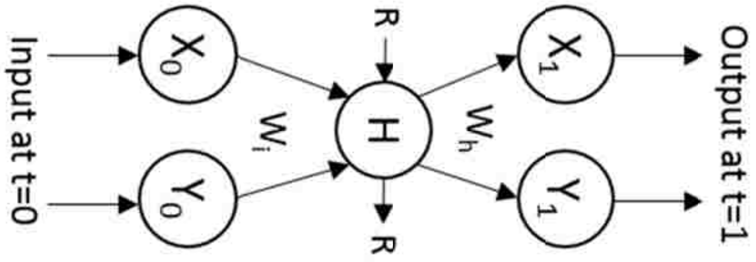


Figure 2-2: A recurrent neural network where the output of the hidden neuron is fed back as an input to the same neuron.

The backwards loops in a recurrent network can be removed by unrolling the network. Although it then appears to be a regular deep network, the weights of every instance of the network are identical.

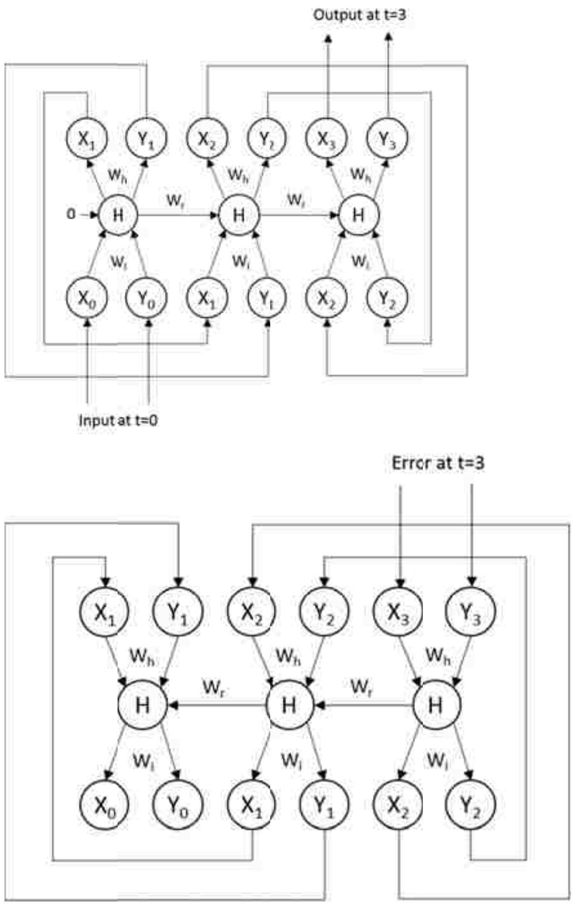


Figure 2-3: On the left, a recurrent network unrolled. This represents Forward Propagation Through Time. On the right, the backwards propagation of error through the same network.

Backpropagation Through Time involves unfolding the network in time until all cycles are removed, then applying normal backpropagation [5].

2.1.3 Deep Learning

Deep learning is a broad category of learning methods that involve the use of networks with large number of layers of hidden nodes. This encompasses Long Short Term Memory, Autoencoder networks, and Convolutional Neural Networks.

2.1.3.1 Autoencoder Networks

Autoencoder networks, or auto-associative networks, use unsupervised learning to compute a lower dimensional representation of the input data. This is accomplished by constructing a network with the same number of outputs as inputs, fewer hidden nodes than input nodes, and target values the same as the input values. This network architecture was first described by Rumelhart et al in 1986 [8].

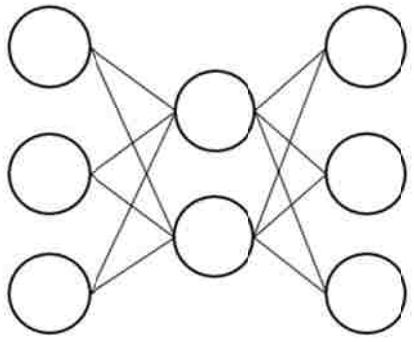


Figure 2-4: An autoencoder network.

Normal backpropagation can be used to train the network. After training the network, the output layer is removed from the network. This leaves two layers – the input layer and the hidden layer. The output of the hidden layer is the lower dimensional representation of the input data.

2.1.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are modelled after the visual cortex's local perceptive fields and deep neural networks for object recognition and classification. This work began with Fukushima's Neocognitron in 1980 [9] and was improved by LeCun in 1998 [10] and Behnke in 2003 [11].

CNNs are typically organized into alternating convolutional layers followed by maxpooling (subsampling) layers. These layers are finally followed by one or more fully connected layers.

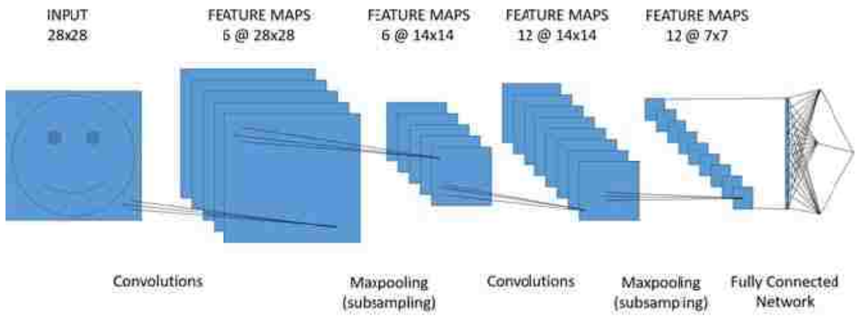
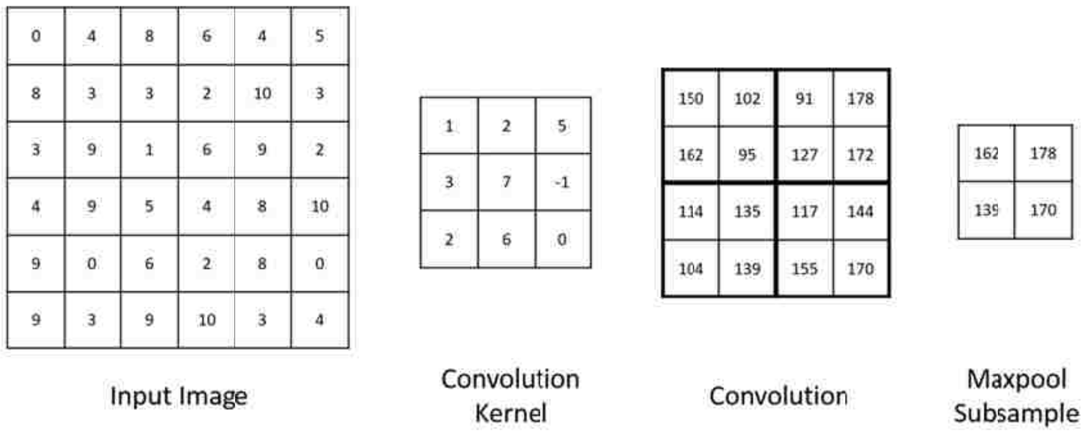


Figure 2-5: A convolutional neural network, similar to LeNet-5 [10], used for object recognition in images.

Convolutional layers are composed of a set of learned convolution kernels that are applied to the output of the previous layer. Because these kernels are small sets of weights applied repeatedly across the entire image, the number of weights to learn is very small, especially compared to a fully connected network.



Input Image

0	4	8	6	4	5
8	3	3	2	10	3
3	9	1	6	9	2
4	9	5	4	8	10
9	0	6	2	8	0
9	3	9	10	3	4

Convolution Kernel

1	2	5
3	7	-1
2	6	0

Convolution

150	102	91	178
162	95	127	172
114	135	117	144
104	139	155	170

Maxpool Subsample

162	178
139	170

Figure 2-6: An example of a convolution followed by a maxpooling operation.

Figure 2-6 shows an example of a convolution kernel applied to an input image. The first pixel of the convoluted image is 150. This is computed using the input image and convolution kernel as a weighted sum... $0\times1+4\times2+8\times5+8\times3+3\times7+3\times-1+3\times2+9\times6+1\times0$.

Maxpooling accomplishes downsampling by dividing the image into regions, then taking the maximum value from each region. A 2x2 maxpool effectively reduces the size of the feature map by 75%.

Because a maxpooling operation produces the same output no matter where in the region the max value was found, a cascade of several maxpooling operations provides some limited shift invariance [10].

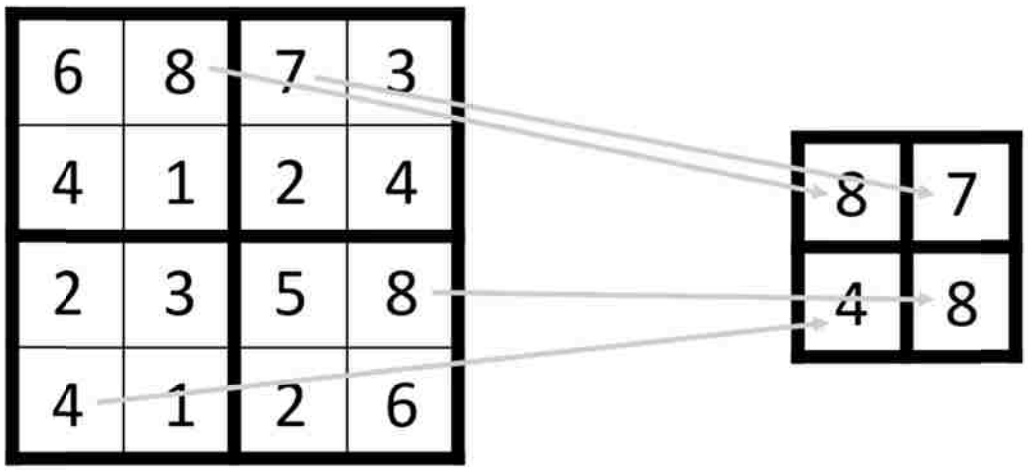


Figure 2-7: An example of a 2x2 maxpooling computation. The image is split into 2x2 regions, and the maximum value from each region is saved. This results in an output image that is ¼ the size of the original.

Several alternating convolution and maxpooling layers produce a large number of small feature maps. These small feature maps are fed into a fully connected neural network.

While a 32x32 input image from the MNIST handwritten digit database with 100 hidden units and 10 output nodes would require 103,400 weights for a fully connected network, a convolutional layer with 100 5x5 kernels would only require 2,500 weights. Even with 10 such layers and a fully connected layer with 1000 hidden nodes and 10 output nodes at the end, the CNN would only require 35,000 learned weights. Due to the action of the maxpool layers, the number of weights actually required much lower.

Additionally, it is common to use a ReLU (rectified linear unit) activation function instead of sigmoid or hyperbolic tangent because of the significantly faster learning rate [12]:

$$\mathrm{f(a)} = \max(0, \mathrm{a}) \tag{2-6}$$

2.1.4 Neural Networks for Forecasting Weather

2.1.4.1 Single location models

Zakerinia et al. developed a neural network to create a wind forecast for a single site using 3 inputs, 20 hidden nodes, and 1 output node that represented the 1 hour wind forecast [13]. Corne et al. also developed a neural network to forecast wind speed for a single site, but used 7 input variables (cloud cover, humidity, pressure, temperature, visibility, wind speed, and wind direction) for the single site. They tested using the 7 variables as inputs, the 7 variables plus 7 more from an hour before, and the 7 variables plus their 1 hour deltas [14]. Abdel-Aal et al. used abductive networks to create a 24-hour hourly temperature forecast. The inputs to the network were temperatures for the 24 previous hours, minimum and maximum temperature for the previous day, and the minimum and maximum forecasted temperature. The output is the temperature for a given hour on the following day [15].

Abistado et al created a forecast for a single location (PAG-ASA Mactan-Cebu Station) using mean dew point, minimum temperature, mean temperature, mean humidity, rainfall, mean wind speed, prevailing wind direction, mean cloudiness, month of year, day of month, and mean pressure as inputs. The output was tomorrow's temperature, humidity, and amount of rainfall [16].

Mao et al created a 24 hour wind power forecast using a neural network with wind speed, wind direction, temperature, humidity, and pressure as inputs [17].

El-Feghi et al used radial basis functions to forecast temperature for a single location (Misrata, Libya). They used humidity, dew point, wind speed, wind direction, and pressure as inputs [18].

Raza and Jothiprakash used data for a single location (Tirunelveli, Tamil Nadu, India) to train a neural network to predict tomorrow's maximum temperature, minimum temperature, humidity, wind speed, sunshine duration, dew point, and evaporation. Their model takes these 7 variables for today and predicts the value of these variables tomorrow [19].

Hayati and Mohela used data from a single site (Kermanshah, Iran) to produce a one-day forecast for tomorrow's high temperature. They used wind, humidity, wet bulb temperature, dry bulb temperature, pressure, sunshine, and radiation as input [20].

Nurcahyo et al predicted rainfall for a single location (Kemayoran Jakarta) using temperature, wind speed, sunshine duration, pressure, humidity, and previous day rainfall as input [21]. Baboo and Shereef used data for a single site (Chhatrapati Shivaji International Airport). The used pressure, temperature, humidity, wind velocity, and wind direction as inputs [22].

There have been many attempts to create neural networks to forecast weather for a single location. However, none of these approaches take into account spatial information, and therefore cannot account for advection and the movement of cyclones and frontal boundaries. For this reason, they have limited use.

2.1.4.2 Multiple location models

There are very few studies that include multiple locations. Collins and Tissot trained 286 neural networks to detect the presence of thunderstorms on a 14x23 grid [23]. Each neural network predicted the presence of a thunderstorm within one box.

This previous work is different from the proposed method in that the former trains a different network for every grid point, where the latter uses the same trained network for every grid point on a given layer. This represents a significant reduction in the number of weights. Additionally, Collins and Tissot's work downscales output from the Eta model. The present work predicts the future values of the input variables.

2.2 Noisy Data, Data Assimilation, and Kalman Filter

Sensor data is noisy. Many algorithms, such as Barnes [24] and Cressman [25] analysis have been developed to solve this problem. These schemes use successive corrections across multiple passes to converge on an estimated denoised grid by applying differing weights to observations of different distances from grid points.

Alternately, if we have a series of data that is observed over time, a Kalman filter [26] may be more appropriate. This is actually how our initialization data is produced by NCEP. Our initialization data is actually a weighted average of a previous forecast plus any new observations [27]. However, because the initialization data is a weighted average of the previous forecast and new data, it's also biased toward the previous forecast and therefore cannot be used to produce an accurate representation of model error [27].

All data used in the experiments described in the present work have already been previously heavily filtered and denoised. Therefore, noise is not considered.

2.3 Choice of Neural Network or Genetic Algorithm for Learning

Choosing which machine learning technique to apply to a given problem is difficult. For the present work of forecasting the ocean and weather, a neural network architecture with backpropagation was chosen.

There are a few ways a genetic algorithm could conceivably be used to solve the same problem. If the governing equations for forecasting the atmosphere could be described by a string/chromosome, then a genetic algorithm could be used to optimize the system. However, describing all the necessary equations as chromosomes would be difficult.

Alternately, a neural network could be used with weights that are learned using a genetic algorithm. However, since we can calculate the error gradient and update the weights using gradient descent, it's better to use backpropagation than a genetic algorithm for optimization.

Genetic algorithms are better used when no gradient can be calculated and only a fitness value can be calculated.

2.4 Numerical Weather Prediction

Numerical Weather Prediction has a long history, beginning with Vilhelm Bjerknes' equations (1904) and Lewis Fry Richardson's failed forecast for May 10, 1910 (1922) in which he computed the entire forecast by hand using his finite difference method and a simplified version of Bjerknes' equations [28].

It's a computationally expensive task that requires not only the numerical solution to a complex set of non-linear partial differential equations (PDEs), but also the creation of a parameterization scheme to estimate sub-grid scale phenomenon [29].

Models such as the NCEP's RAP (Rapid Refresh) and HRRR (High Resolution Rapid Refresh) model use this technique to generate forecasts.

2.4.1 The Primitive Equations

The Primitive Equations are a set of non-linear partial differential equations that govern atmospheric physics. A variation of these equations for the backbone of a modern dynamical numerical weather prediction model.

Horizontal motion equations [29]:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - \omega\frac{\partial u}{\partial p} + fv - \frac{\partial \phi}{\partial x} + F_x \qquad 2\text{-}7$$

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - \omega\frac{\partial v}{\partial p} - fv - \frac{\partial \phi}{\partial y} + F_y \qquad 2\text{-}8$$

Thermodynamic temperature equation [29]:

$$\frac{\partial T}{\partial t} = -u\frac{\partial v}{\partial t} - v\frac{\partial v}{\partial t} + \omega\left(\frac{RT}{C_p p} - \frac{\partial T}{\partial P}\right) + \frac{H}{C_p} \qquad 2\text{-}9$$

Conservation of moisture equation [29]:

$$\frac{\partial q}{\partial t} = -u\frac{\partial q}{\partial x} - v\frac{\partial q}{\partial y} - \omega\frac{\partial q}{\partial p} + E - P \qquad 2\text{-}10$$

Conservation of mass (continuity equation) [29]:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial p}=0 \quad 2\text{-}11$$

Hydrostatic equation [29]:

$$\frac{\partial \phi}{\partial p}=-\frac{RT}{p} \quad 2\text{-}12$$

Table 2-1: List of variables and their meaning.

Variable	Meaning
u	East-West Component of the Wind
v	North-South Component of the Wind
ω	Vertical Velocity
T	Temperature
q	Specific Humidity
φ	Geopotential Height
p	Pressure

These equations are not complete without an additional set of equations (parameterizations) to estimate the value of F_x, F_y, H, E, and P. These correspond to friction in the x and y directions, heat, evaporation, and precipitation. The heating term corresponds to heating due to incoming solar radiation, outgoing terrestrial radiation, latent heat of condensation, latent heat of vaporization, and latent heat of fusion.

2.4.2 Finite Difference Methods

The primitive equations contain many spatial derivatives. These are partial derivatives with respect to the east-west and north-south grid coordinates x and y. They are computed numerically using a centered finite difference scheme:

$$\frac{\partial T_{ij}}{\partial x}=\frac{T_{i+1,j}-T_{i-1,j}}{2} \quad 2\text{-}13$$

$$\frac{\partial T_{ij}}{\partial y}=\frac{T_{i,j+1}-T_{i,j-1}}{2} \quad 2\text{-}14$$

This process is similar to an edge detection convolution kernel that detect vertical and horizontal edges, respectively.

2.4.3 Time Integration

The learning task of our recurrent convolutional neural network is to learn to compute the partial derivative of each variable with respect to time. For temperature, this would be $\frac{\partial T_{ij}}{\partial t}$. Once this is known, we can time step forward to get the next value of T:

$$T_1 = T_0 + \Delta t \frac{\partial T_0}{\partial t} \tag{2-15}$$

$$T_{t+1} = T_{t-1} + 2 \times \Delta t \frac{\partial T_t}{\partial t} \tag{2-16}$$

The first formula is a forward integration technique, while the second is a centered-in-time technique, or leapfrog. While we could use the first one for every time step, errors quickly ruin the forecast unless a very small time step is used [29]. It was confirmed experimentally (as part of the present work) with a 15 second time step that the forward integration technique underperforms the leapfrog scheme using only a 5-minute time step. For this reason, we only use the forward scheme in the first time step to get the leapfrog scheme started.

2.4.4 The CFL Condition

We are using 20km resolution input data and 1 hour later target values. Ideally, we would take that input data and create a 1 hour forecast. However, it was discovered by Courant, Friedrichs, and Lewy that forecast stability is a function of grid resolution, time step, and velocity [30].

$$C = \frac{u_{max} \Delta t}{\Delta x} \leq C_{max} \tag{2-17}$$

The ideal value of C_{max} depends on many factors, including the system solution method. For our purposes, we'll take it to be equal to 1. This means that with a 1-hour time step and 20 km grid spacing, the maximum wind velocity we can simulate without the simulation becoming unstable is approximately 5.5 m/s or 12 mph. This is much lower than the typical maximum wind speed, even at the surface. Jet streams and cyclones can have wind speeds that exceed 150 mph.

If we change our time step to 6 minutes and keep the same grid spacing, we can simulate wind speeds up to 55.5 m/s, or 124 mph. This necessarily smaller time step makes designing our system much more difficult because we don't have target values for only 6 minute later. It also means that the forecast system must run for 10 iterations in order to create a 1 hour forecast, increasing computation time by a factor of 10.

For reference, the RAP model uses a 1-minute time step. This allows for very high wind speeds in a very stable model.

2.4.5 Parameterization Schemes

Parameterization schemes are used to estimate sub-grid scales phenomena that can't be directly simulated [31]. Krasnopolsky et al. replaced the shortwave and longwave atmospheric radiation parameterization schemes of the NCAR CAM-2 model with a neural network. The network proved to be a fast and accurate replacement and resulted in a 50-80 times faster computation of the radiation parameterization [32].

2.4.6 Rapid Refresh (RAP) Model and the Weather Research and Forecasting (WRF) Model

The NCEP RAP model is a specially configured version of the WRF model. It's setup to create an 18 hour forecast for the continental United States in 1 hour increments.

It uses WRF version 3.6.1 with ARW Core. The physic suite includes Grell-G3 convection, Thompson/NCAR microphysics, RRTM longwave radiation, Goddard shortwave radiation, MYNN-Olson turbulent mixing, and RUC-Smirnova land-surface model [33].

NCEP runs three versions of the RAP model, at 20km, 13km, and 3km resolutions.

2.5 Parallelization and Scalability

2.5.1 Motivations

Forecasting is a very time-sensitive task. Even if a forecast is 100% accurate, it's useless if the forecast takes too long to produce.

Advances in accuracy come with increased model resolution. Unfortunately, halving the resolution increases the number of grid points, and therefore computation time, by a factor of four. In accordance with the CFL condition (section 2.2.4), a decrease in spatial resolution also necessitates a decrease in time step. This further increases the computational complexity.

In order to quickly produce a forecast for a larger region, we must introduce parallelism.

2.5.2 OpenMP

OpenMP is a framework for creating parallel code on a single machine by spreading computation across processors. This is done by adding #pragma preprocessor directives to the code. For instance:

```
Matrix<float> transpose(Matrix<float> m){
    Matrix<float> m2(m.cols, m.rows);
    #pragma omp parallel for
    for(int i = 0; i < m.rows; i++)
        for(int j = 0; j < m.cols; j++)
            m2[j][i] = m[i][j];
    return m2;
}
```

The code above computes the transpose of a matrix. Line 3 contains the #pragma OpenMP command. This line simply spreads the work of the for loop across as many processors as available. In this example, each processor will compute the transpose of a row. Because there are no dependencies between loop iterations, the output of the parallel version is the same as the output of the sequential version.

It's also possible to do reduction operations. The code below computes the sum of all elements of a matrix:

```
float sum(Matrix<T> m){
    float sum = 0;
    #pragma omp parallel for reduction (+ : sum)
    for(int i = 0; i < m.rows; i++)
        for(int j = 0; j < m.cols; j++)
            sum += m.data[i*m.cols+j];
    return sum;
}
```

Line 3 of this code specifies that OpenMP should be used to parallelize this loop, that each processor should have its own separate variable named sum, and that the reduction operator should be + (summation). After the loop, all the different values for sum are added together and execution resumes just as if the code had been executed sequentially.

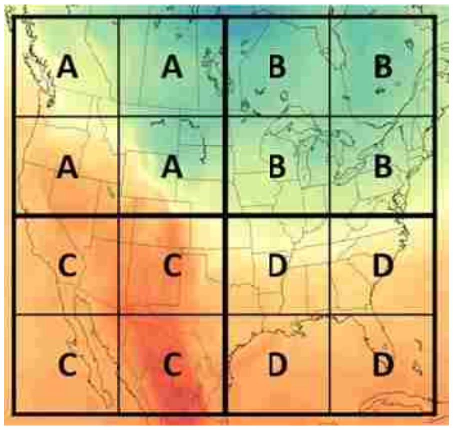


Figure 2-8: Above is a 4x4 grid. A single processor could compute the forecast for all 16 locations, or 4 processors could be used in parallel to each compute one of the four quadrants. This would lead to a 75% reduction in computation time.

Part of writing the neural network and backpropagation code involved writing a library for efficient matrix math operations. Adding OpenMP to the matrix library significantly improved performance.

2.6 Summary

Previous work in this direction has been focused mainly on either forecasting weather variables for a single location and learn using inputs from only that site, or focused on creating a hybrid dynamic climate model by applying machine learning to the parameterization scheme. The former ignores the important spatial component that is available and essential to a successful forecast, while the latter hybrid model only partially relies on machine learning. For this reason, the method proposed is a generalized recurrent convolutional differentiation-integration neural network that utilizes both spatial and temporal information to generate a forecast for a wide region. Instead of developing a hybrid model, the method almost exclusively relies on learning with limited domain knowledge.

The learning method presented is an extended form of Backpropagation Through Time for a recurrent network with outputs that feed back through as inputs only after undergoing a fixed transformation.

3 METHODOLOGY

3.1 INTRODUCTION

In this chapter, we introduce the new concepts we have developed. In particular, we introduce the Differentiation-Integration Time Step Network and discuss how to train it and parallelize it to speed up computation.

3.2 DIFFERENTIATION-INTEGRATION TIME STEP NETWORK

3.2.1 DITS Network Architecture

A Differentiation-Integration Time Step (DITS) Network is a network composed of two parts – a traditional neural network layer, and a weighted summation layer.

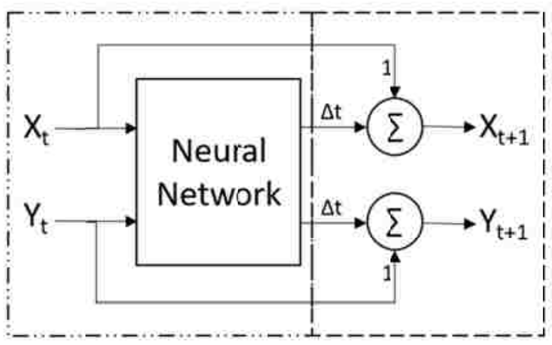


Figure 3-1: A Forward Integration DITS Network

The neural network in the left half of Figure 3-1 is used to compute the time derivatives ($\frac{\partial X_0}{\partial t}$ and $\frac{\partial Y}{\partial t}$) of the input variables. The weighted summation on the right forward integrates the input variables to the next time step. This corresponds to the equation 2-14.

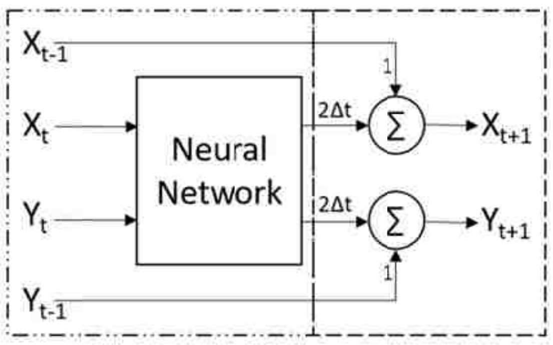


Figure 3-2: A Centered-In-Time DITS Network

Figure 3-2 depicts a centered-in-time (leapfrog) DITS network, which corresponds to equation 2-14. This architecture produces a significant increase in numerical stability and accuracy over Figure 3-1. The main difference is the addition of X_{t-1} and Y_{t-1} as inputs and the time step coefficient change from Δt to $2\Delta t$.

3.2.2 Backpropagation of Error

Error is computed at the output of the network. To back propagate error through Figure 4, let's look at the equation in the second half:

$$X_1 = X_0 + \Delta t \frac{\partial X_0}{\partial t} \qquad 3\text{-}1$$

Error in X_1 can be attributed to two sources, X_0 and $\Delta t \frac{\partial X_0}{\partial t}$. If we assume that the initial state X_0 is accurate, all error comes from the second half of the equation. Δt is a constant, so all error comes from $\frac{\partial X_0}{\partial t}$. Therefore, the error we back propagate through the neural network in the left side of the DITS network is $\frac{1}{\Delta t}\frac{\partial X_0}{\partial t}$. This is done using the standard backpropagation network.

3.3 RECURRENT CONVOLUTIONAL DITS NETWORK

3.3.1 Motivations

While an ordinary DITS network is appropriate for a single time step, a recurrent version is needed for forecasting multiple time steps into the future. The convolutional portion refers to using a single set of weights across the entire input to create a feature map. In this instance, the feature map is a 2D map of time derivatives. This allows us to incorporate spatial information.

3.3.2 Network Architecture

This type of network is conceptually similar to a convolutional neural network. CNNs typically consist of alternating convolution and maxpooling layers. This network consists of alternating convolution and integration layers. However, the 'convolution kernel' is a 3-layer neural network instead of a traditional n x n set of weights. This is done because multilayer neural networks can be used as universal approximators.

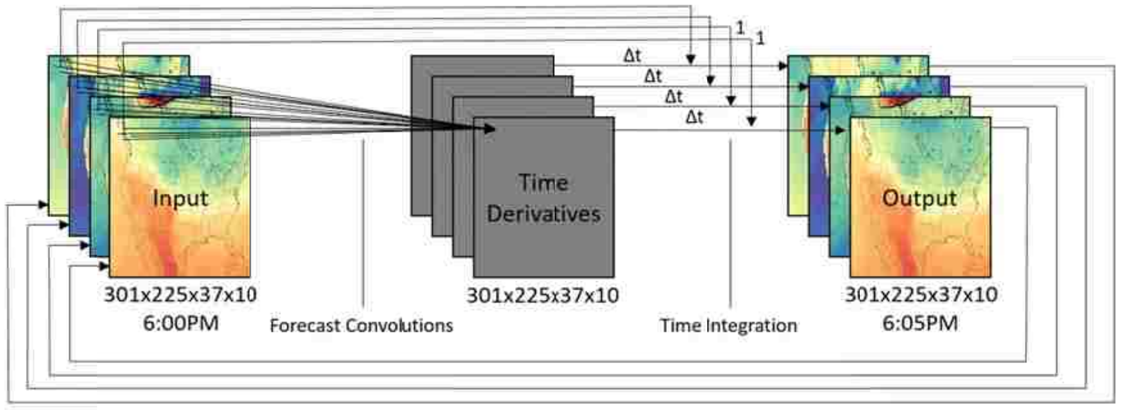


Figure 3-3: A Recurrent Convolutional DITS Network. On the left, each plane represents a 2D input variable, such as temperature, humidity, or geopotential height. In the center we have the derivatives of each of the input variables with respect to time. At the end, we have the output variables one time-step in the future.

The key assumption is that forecasting is location shift invariant. The same physics apply everywhere, so the learned convolutional weights apply to the entire image. This represents a significant reduction in the number of weights that must be learned.

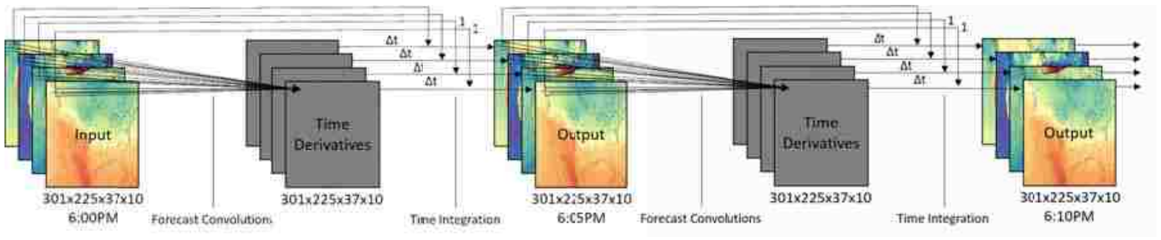


Figure 3-4: Two applications of the Recurrent Convolutional DITS Network from Figure 3-3 unrolled to show a forecast two time steps into the future.

Just as the application of a convolution kernel to an image in a CNN produces a feature map, the application of the shared weight neural network across our input produces a map of time derivatives.

3.3.3 Backpropagation of Error in Time

First, we unroll the network to remove loops. Again, error is computed at the output of the network. Let's look at the equation driving the time integration in Figure 3-3:

$$X_{t+1} = X_t + \Delta t \frac{\partial X_t}{\partial t} \qquad 3\text{-}2$$

Error in X_{t+1} can be attributed to error in X_t and $\Delta t \frac{\partial X_t}{\partial t}$. Unlike the non-recurrent in Section 3.2.1, we cannot assume the error in X_t is 0.

Therefore, we must come up with a rule to attribute a portion of the error to each in terms of the error in the output, which is known:

$$\mathrm{Error}_{X_t} = \lambda\,\mathrm{Error}_{\mathrm{output}} \tag{3-3}$$

$$\mathrm{Error}_{\Delta t\frac{\partial X_t}{\partial t}} = (1-\lambda)\,\mathrm{Error}_{\mathrm{output}} \tag{3-4}$$

$$0 \le \lambda \le 1 \tag{3-5}$$

The parameter λ can decay exponentially with time. In the last application (going backward), X_0 should be assumed to be accurate and therefore λ is 0. This is equivalent to the method used in Section 3.1.2.

For instance:

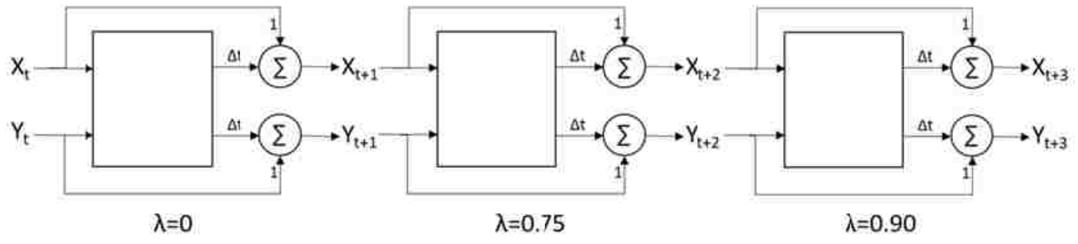


Figure 3-5: Three applications of a DITS network with a varying lambda.

Above, we have 3 applications of a DITS network to forecast values for X_1, Y_1, X_2, Y_2, X_3, Y_3, given initial values $X_{t=0}$, $Y_{t=0}$.

If error in the output (Error_{X_3}) is 10 and λ is 0.9, then:

$$\mathrm{Error}_{X_2} = 0.9(10) = 9 \tag{3-6}$$

$$\mathrm{Error}_{\Delta t\frac{\partial X_2}{\partial t}} = (1-0.9)(10) \tag{3-7}$$

If $\mathrm{Error}_{\Delta t\frac{\partial X_2}{\partial t}}$ is 1, then because Δt is a constant:

$$\mathrm{Error}_{\frac{\partial X_2}{\partial t}} = \frac{1}{\Delta t}\mathrm{Error}_{\Delta t\frac{\partial X_2}{\partial t}} \tag{3-8}$$

This value $\mathrm{Error}_{\frac{\partial X_2}{\partial t}}$ is the error backpropagated through the third application of the network using the standard backpropagation algorithm. For the second application of the network:

$$\mathrm{Error}_{X_1} = 0.9(9) = 8.1 \tag{3-9}$$

$$\mathrm{Error}_{\Delta t\frac{\partial X_1}{\partial t}} = (1 - 0.9)(9) = 0.9 \tag{3-10}$$

And for the first application, where λ is always 0:

$$\mathrm{Error}_{X_0} = 0(8.1) = 0 \tag{3-11}$$

$$\mathrm{Error}_{\Delta t\frac{\partial X_0}{\partial t}} = (1 - 0)(8.1) = 8.1 \tag{3-12}$$

This can be coupled with a progressive reduction in time step size during training, where training begins with the time step equal to 1 hour. This can be progressively reduced to 30 minutes, 20 minutes, 15 minutes, 12 minutes, 10 minutes, where each of these phases is as described above.

3.4 Parallelization

3.4.1 Spatial Decomposition

Performance of the forward propagation phase is the most important aspect of the system because that's where most of the work occurs in the everyday running of the system. Once training is complete, the backpropagation portion is no longer used. However, the forward propagation phase is used every single time a forecast is created.

We use OpenMP to parallelize the implementation in order to speed up the computation time.

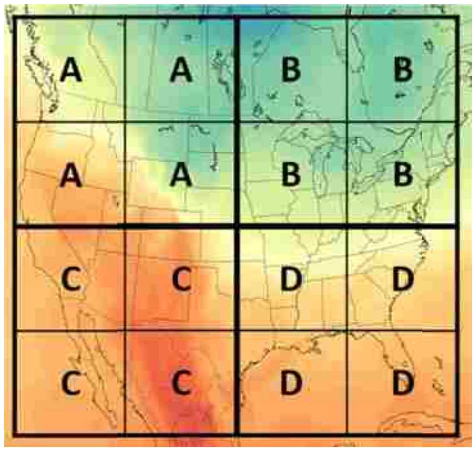


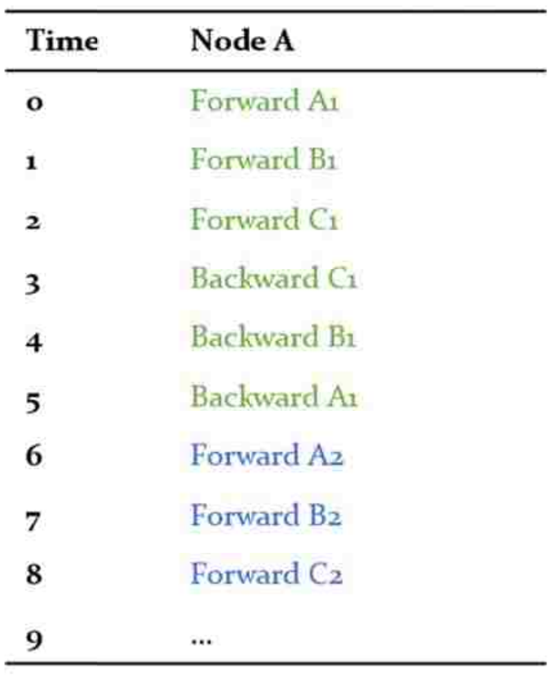
Figure 3-6 Spatial decomposition for parallelization

The grid is broken down in a manner similar to Figure 3-6. Because the work of forecasting for each location independent of forecasting for other locations, it can be allocated to a pool of threads on different processors to do the work in parallel. Every processor (or core) will be given a subgrid (A, B, C, or D) to process.

3.4.2 Pipelined Forward-Backpropagation Through Time

Training a neural network is a sequential process. Before you can update the network's weights, you have to back propagate the error. Before you can compute the error, you have to forward propagate sample input through the network to compute its output.

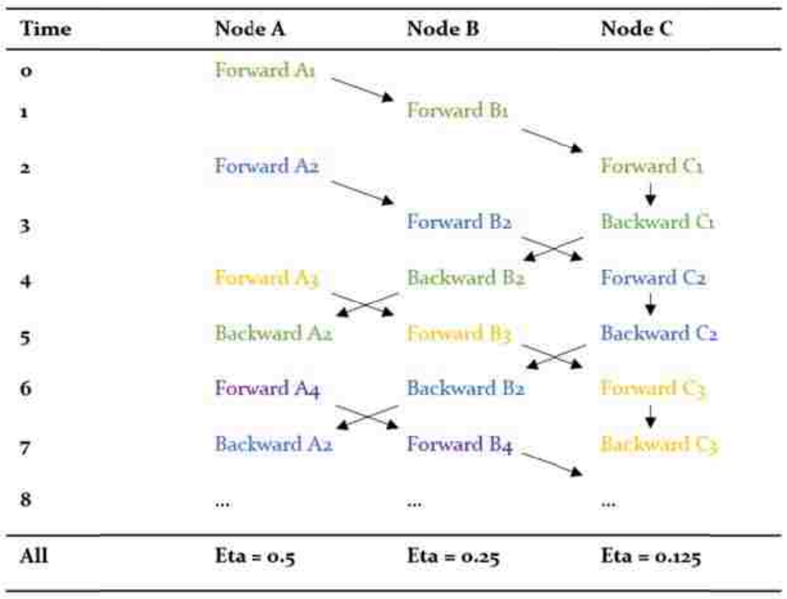
This process is especially evident in deep recurrent neural networks. No learning can be done until the inputs propagate through all the layers to the output. For a recurrent network, this also involves forward propagation through time.



Time	Node A
0	Forward A1
1	Forward B1
2	Forward C1
3	Backward C1
4	Backward B1
5	Backward A1
6	Forward A2
7	Forward B2
8	Forward C2
9	...

Figure 3-7: Forward propagation followed by backward propagation through a network with three layers, or a network recurrently applied three times.

Figure 3-6 shows the forward propagation followed by back propagation through a network with three layers on a single machine (Node A) with no parallelism. It takes 6 time steps before all the weights get updated and the next round can begin.



Time	Node A	Node B	Node C
0	Forward A1		
1		Forward B1	
2	Forward A2		Forward C1
3		Forward B2	Backward C1
4	Forward A3	Backward B2	Forward C2
5	Backward A2	Forward B3	Backward C2
6	Forward A4	Backward B2	Forward C3
7	Backward A2	Forward B4	Backward C3
8	...	...	...
All	Eta = 0.5	Eta = 0.25	Eta = 0.125

Figure 3-8: Pipelined forward and backward propagation across three machines.

Figure 3-7 shows a pipelined version. Here, the learning task is spread across three machines. Each machine is entirely focused on one layer of the network, or one application of a recurrent network. While it takes 4 time steps until the pipeline is full, once it fills it completes a full forward-backward propagation every other time step.

For recurrent applications of the same network, a different learning rate could be used on each machine. This can be seen at the bottom of Figure 3-7.

This method has the disadvantage of significant data transfer overhead. For this reason, the spatial decomposition approach is preferred.

4 Case Study: Ocean Modelling

4.1 Introduction

This chapter describes an implementation of an ocean model using the proposed methodology. In this instance, we have the current values of all variables (temperature, salinity, and velocity in u and v directions) at t=0 on a 350 x 340 x 39 grid, and the estimated ground truth values 3 hours later.

4.2 Motivation

The Loop Current is a warm water current that flows into the Gulf of Mexico through the Yucatan Channel (between the Yucatan Peninsula and Cuba) and back out through the Florida Straits (between Cuba and Florida).

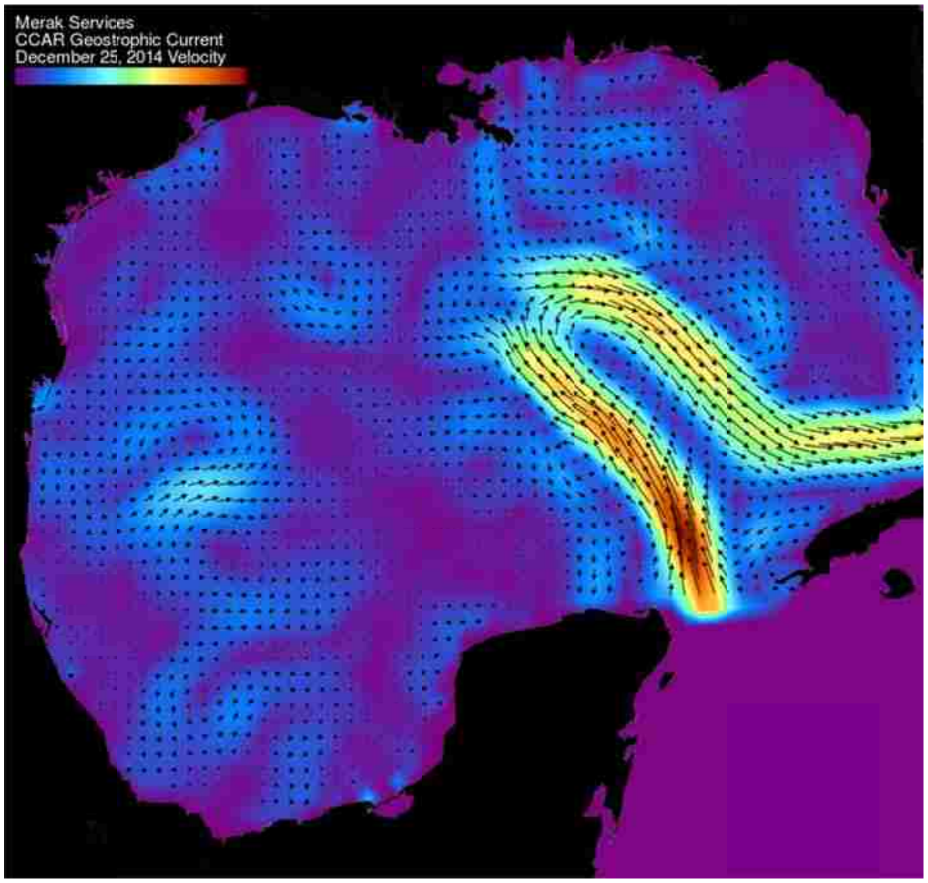


Figure 4-1: Surface water current velocity vector field of the Gulf of Mexico. The Loop Current is the dominating feature in the Eastern half of the Gulf. Data provided by CCAR and Dr. Robert Leben.

The Loop Current is the dominant feature in the eastern Gulf of Mexico. The current slowly extends further and further north towards the northern Gulf coast over a period of 6 to 18 months [2]. Eventually, the current short circuits and changes to a far more southerly course, leaving the northern portion to form a warm core eddy. This newly formed eddy slowly drifts to the west until it encounters friction from the Texas coast and dissipates.

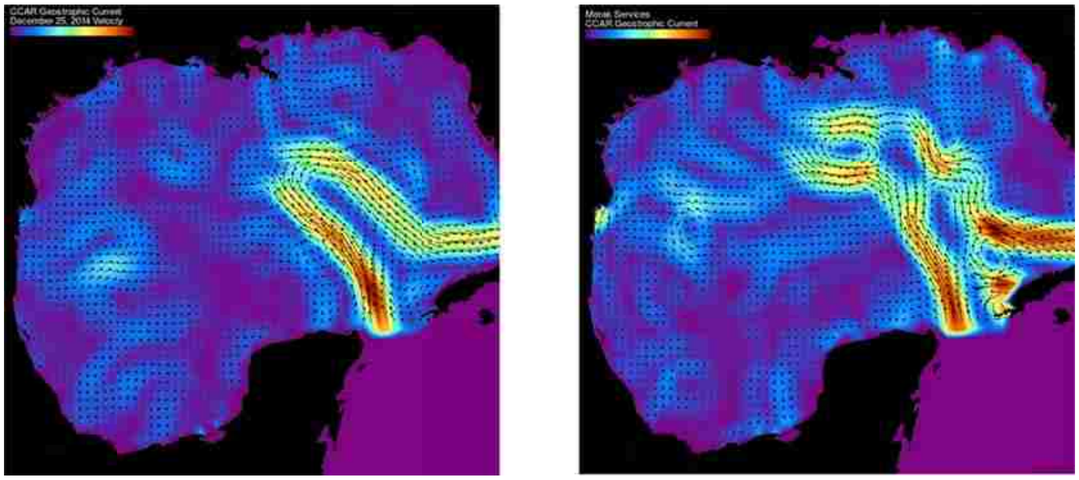


Figure 4-2: On the left, a typical water current image of the Loop Current in the Gulf of Mexico. On the right, a very erratic Loop Current extends significantly further north. Data provided by CCAR and Dr. Robert Leben.

Occasionally, this process can cause significant currents of up to 5 knots to encounter deep water oil drilling rigs off the coast of Louisiana. Such occurrences necessarily result in downtime and can cause significant damage. This has happened as recently as late last year [3].

Additionally, it has been found that hurricanes that traverse over the Loop Current, or warm core eddies, tend to quickly strengthen into powerful storms. However, hurricanes that pass over cold core frontal eddies tend to weaken [34].

For these reasons, forecasting this current and other like it would be very useful.

4.3 LOM: Learned Ocean Model

4.3.1 Introduction

The proposed method was implemented in Python and C++. All neural network code was written by the author specifically for this task. The experiment was run on a laptop with a 2.4GHz Intel Core i5-6300U processor and 8GB RAM. Training time was limited to 1-day total for all networks, but could be allowed to run longer for reduced error. Because water current (u and v components), temperature, and salinity was forecast on 39 levels of the ocean, this required training 156 different networks.

4.3.2 Data

The network was trained using the input data sets to the Navy Coastal Ocean Model (NCOM) for March 1, 10, and 20th of 2016, and validated against March 15th.

The NCOM model over the American Mediterranean Sea is run daily out to 96 hours in 3 hour intervals on an 814x1294 grid with a 3km horizontal resolution and 40 vertical levels with depth coordinates from 0m to 5000m. Data includes temperature, water current speed/direction, temperature, and salinity for the entire water column. This data can be downloaded from NCEP [33].

Because the same learned network is applied to every grid point at a given depth to create a forecast, interaction with land therefore needs to be taken into account. This effect is reduced by selecting a 350x340 sub-grid over the Atlantic Ocean.

Training and validation data was generated by computing input data for every grid point and generating target values for a 3 hour forecast by using the output files for the NCOM model for 3 hours later.

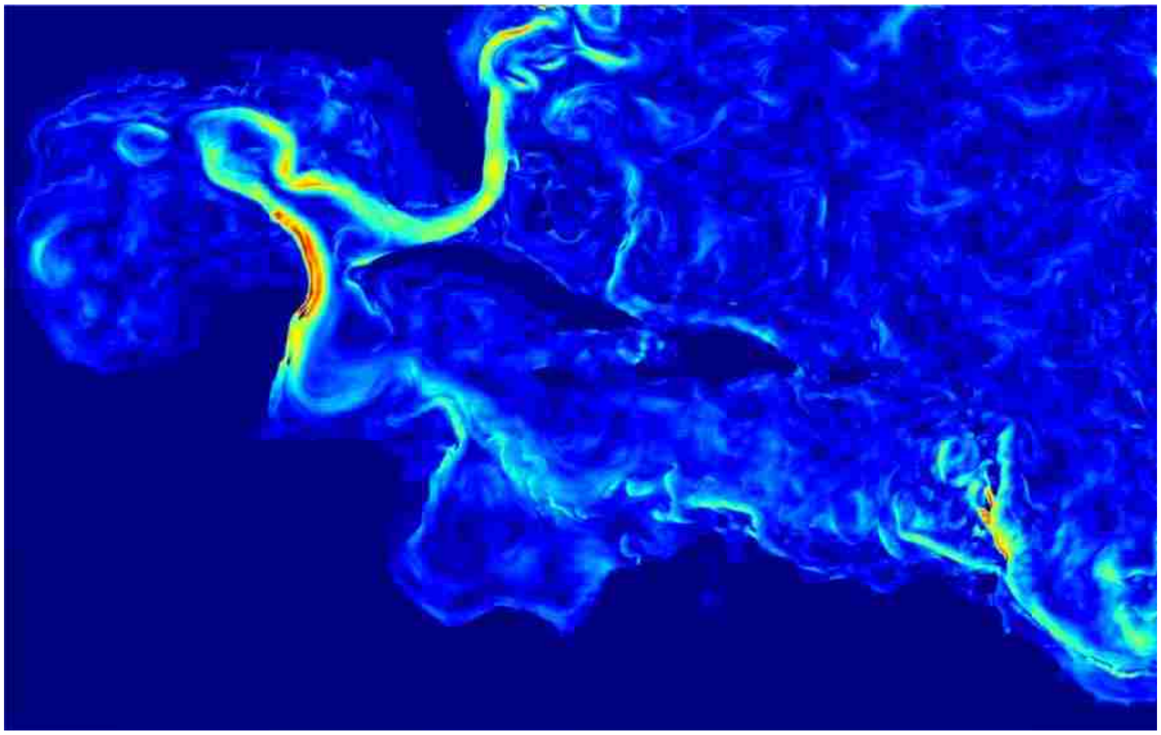
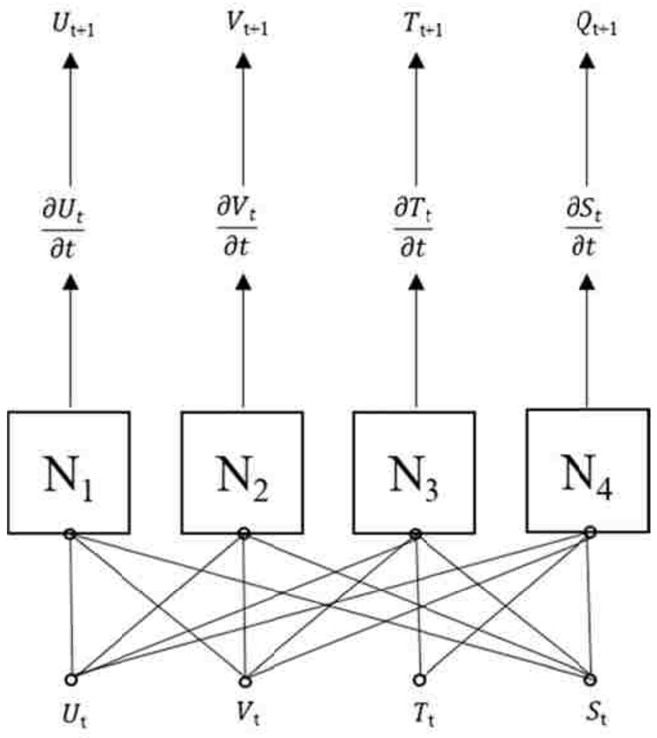


Figure 4-3: NCOM input data depicting water current velocity for the Caribbean Sea and Gulf of Mexico.

4.3.3 Architecture

The architecture is identical to that of Figure 5-2, except that the only input variables are temperature, salinity, and the water current in the u (east-west) and v (north-south) directions.



U_{t+1}
V_{t+1}
T_{t+1}
Q_{t+1}
$\frac{\partial U_t}{\partial t}$
$\frac{\partial V_t}{\partial t}$
$\frac{\partial T_t}{\partial t}$
$\frac{\partial S_t}{\partial t}$
N_1
N_2
N_3
N_4
U_t
V_t
T_t
S_t

4.4 Analysis

Error was measured as the Mean Absolute Error (MAE):

$$\mathbf{MAE} = \frac{\mathbf{1}}{\mathbf{n}} \Sigma_{\mathbf{ij}} \left| \mathbf{t_{ij}} - \mathbf{y_{ij}} \right| \tag{4-1}$$

The forecasts generated by the proposed approach were compared to forecasts generated by the NCOM model, persistence, and climatology. The MAE of the 1 hour forecast generated by each model is calculated for the same 350 x 340 x 39 sub-grid.

4.5 Results

4.5.1 Computational Complexity of LOM vs NCOM

LOM works on a 350 x 340 x 39 grid (or 4,641,000) and takes approximately 5.8 seconds to compute one time-step on a Surface Book laptop.

NCOM works on a 1120 x 778 x 36 grid (or 31,368,960 grid points) and takes approximately 35 minutes per model day on 24 Intel Cores and a 4 minute time-step [27].

If NCOM were run on our 350 x 340 x 39 grid, assuming linear scaling, it would take approximately 20.7 seconds.

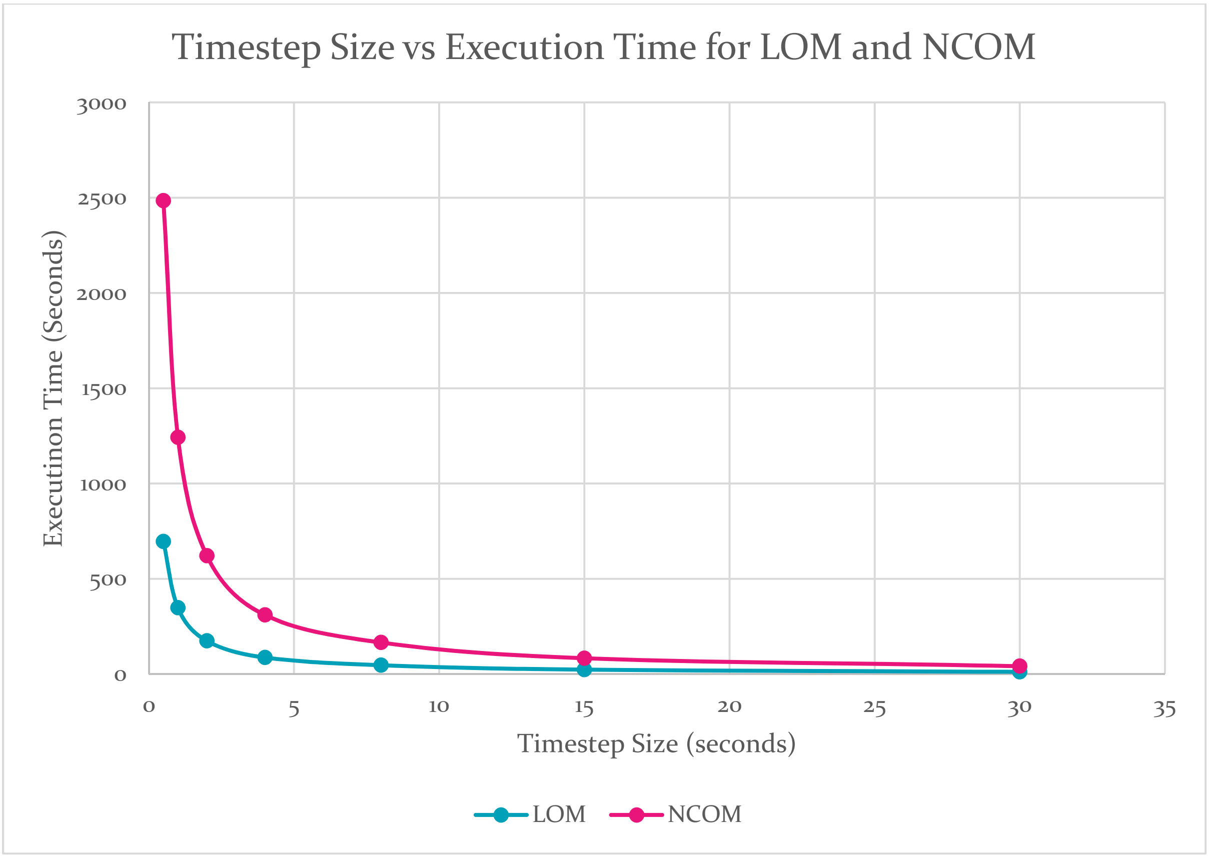


Figure 4-4 Time-step Size vs Execution Time for LOM and NCOM for a 1 hour forecast

Therefore, LOM is approximately 20.7/5.8 = 3.6 times faster, even before considering that the LOM was run on a low power laptop processor, and NCOM is run on likely much faster desktop class processors.

With a nearly factor of four speed up over NCOM, this would allow LOM to work on a grid with twice the horizontal resolution of NCOM with the computational resources currently allocated to NCOM.

Figure 4-4 shows how the execution time of both LOM and NCOM vary with time-step size for a 1 hour forecast. The smaller the time-step, the more pronounced the difference in execution time becomes.

Although LOM is already very fast, significant optimizations can be made to further speed it up. A significant amount of time is wasted copying data to ready it for the neural network. If the matrix and neural network library were re-written, LOM (and LM3) would be significantly faster.

4.5.2 LOM Salinity Forecasting

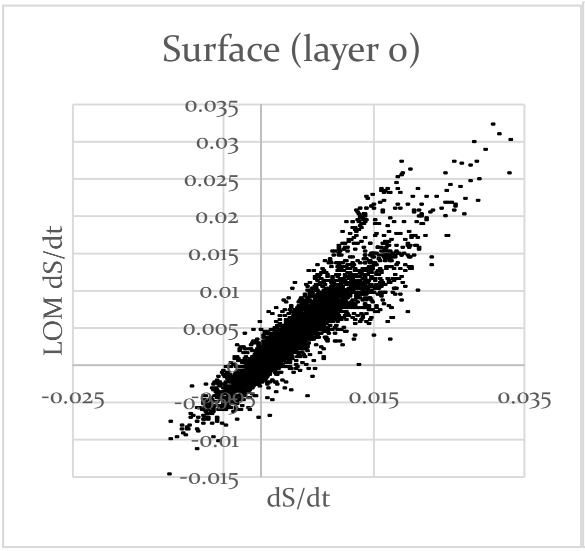
Table 4-1 Summary of results forecasting salinity for 8 levels of the ocean, where LOM represents the results of the proposed method. Error is MAE.

Level	LOM Salinity	NCOM Salinity	Persistence Salinity	Climate Salinity
0m	0.013884	0.010804	0.01169	0.14032
10m	0.005487	0.010515	0.011116	0.122367
30m	0.005662	0.010849	0.010631	0.106313
60m	0.005594	0.009166	0.010892	0.096388
125m	0.014748	0.005943	0.022535	0.080334
350m	0.010341	0.004445	0.015618	0.053019
800m	0.011099	0.00461	0.016777	0.034504
2000m	0.001599	0.0002	0.002476	0.029792

LOM performed better than NCOM on the first 16 levels of the ocean below the surface, with an average error for NCOM around 80% higher. In addition to levels 1-16, LOM also outperformed NCOM on level 33 with a 15% difference in error.

Table 4-2 Table of results for levels where LOM performed better than NCOM, persistence, and climatology.

Level	LOM Error Salinity	NCOM Error Salinity
2 m	0.005607	0.010635
4 m	0.005843	0.010575
6 m	0.005624	0.010528
8 m	0.005550	0.010506
10 m	0.005487	0.010515
12 m	0.005384	0.01056
15 m	0.005302	0.010659
20 m	0.005215	0.010856
25 m	0.005210	0.010942
30 m	0.005134	0.010849
35 m	0.005097	0.010614
40 m	0.005095	0.010271
45 m	0.005108	0.009923
50 m	0.005226	0.009637
60 m	0.005594	0.009166
70 m	0.006602	0.008874
1250 m	0.0009348	0.00100144



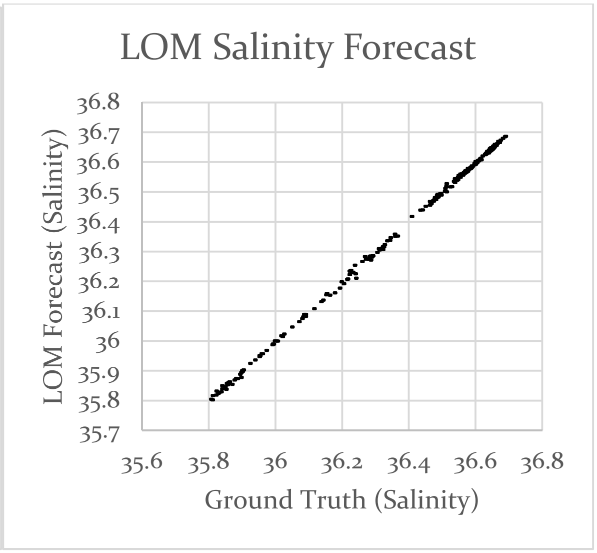


Figure 4-5 LOM salinity forecast at the surface. On the left, a comparison of the forecast change in salinity vs actual change in salinity. On the right, actual salinity vs forecasted salinity.

4.5.3 LOM Water Temperature Forecasting

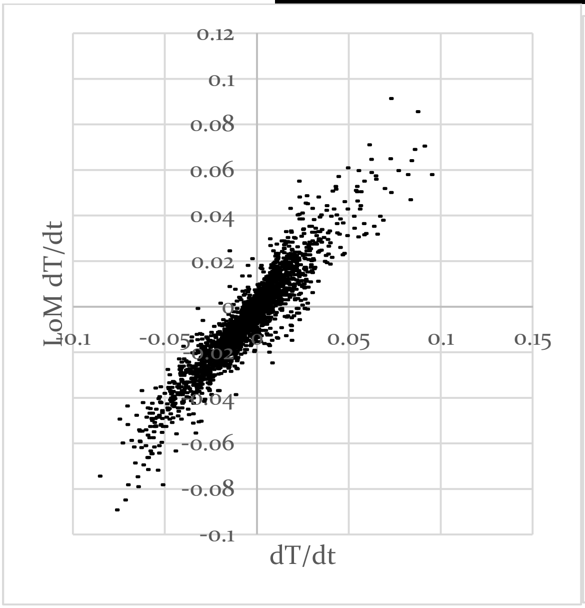
Table 4-3 Summary of results forecasting temperature for 8 levels of the ocean, where LOM represents the results of the proposed method. Error is MAE in Celsius.

Level	LOM Error Temperature	NCOM Error Temperature	Persistence Error Temperature	Climate Error Temperature
0m	0.140178	0.134634	0.130658	0.269382
10m	0.062074	0.138412	0.063263	0.297142
30m	0.050006	0.115152	0.041969	0.296411
60m	0.054653	0.08761	0.041167	0.279829
125m	0.23243	0.059692	0.233179	0.378432
350m	0.103416	0.033024	0.099292	0.147696
800m	0.157645	0.041699	0.161521	0.198037
2000m	0.016453	0.00205	0.024667	0.050963

LOM outperformed NCOM on levels 2 through 16 (4 meters to 80 meters), with NCOM's error from 32% to 134% higher than LOM.

Table 4-4 Table of results for temperature forecasting for levels where LOM performed better than NCOM. Error is MAE.

Level	LOM Error Temperature	NCOM Error Temperature
4 m	0.100867	0.135179
6 m	0.078911	0.136295
8 m	0.068323	0.13773
10 m	0.062074	0.138412
12 m	0.058972	0.138005
15 m	0.057884	0.135585
20 m	0.055215	0.128569
25 m	0.052542	0.121276
30 m	0.050006	0.115152
35 m	0.051132	0.109383
40 m	0.052422	0.103977
45 m	0.053044	0.099151
50 m	0.053429	0.094972
60 m	0.054653	0.08761
70 m	0.060478	0.080166
80 m	0.040964	0.071781



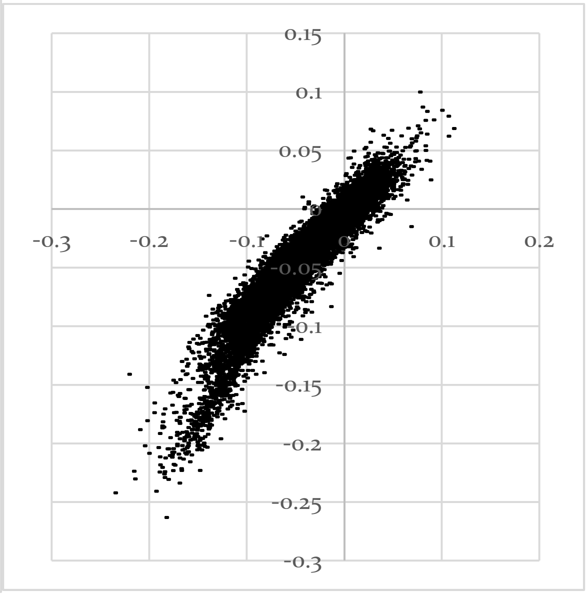


Figure 4-6 LOM Water temperature forecast. On the left, scatterplot of forecasted change in temperature vs actual change in temperature at the surface.

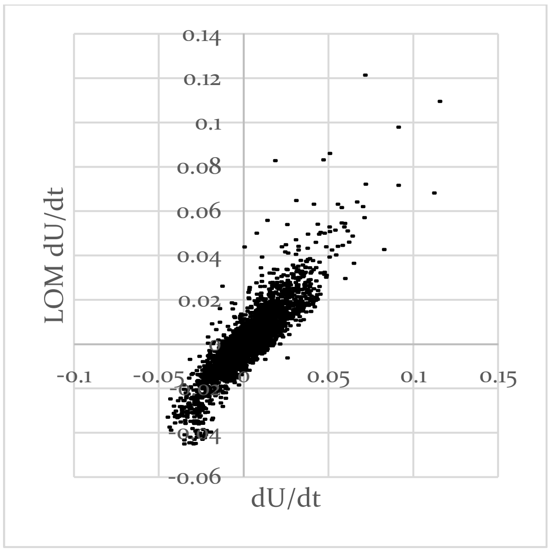
4.5.4 LOM Water Current Forecast

Table 4-5 Summary of results forecasting temperature for 8 levels of the ocean, where LOM represents the results of the proposed method. Error is MAE in Celsius.

Level	LOM	NCOM	Persistence	Climate
	Error U	Error U	Error U	Error U
0m	0.106984	0.017094	0.104862	0.14032
10m	0.099336	0.016522	0.084165	0.122367
30m	0.080445	0.01203	0.07347	0.106313
60m	0.07187	0.010742	0.070892	0.096388
125m	0.055384	0.008957	0.06116	0.080334
350m	0.045684	0.006257	0.046137	0.053019
800m	0.036105	0.002996	0.033173	0.034504
2000m	0.008534	0.00174	0.022524	0.029792

Table 4-6 Summary of results forecasting temperature for 8 levels of the ocean, where LOM represents the results of the proposed method. Error is MAE in Celsius.

Level	LOM	NCOM	Persistence	Climate
	Error V	Error V	Error V	Error V
0m	0.104665	0.019712	0.098368	0.127312
10m	0.100995	0.017163	0.087528	0.12321
30m	0.09404	0.010977	0.076755	0.113392
60m	0.095549	0.010107	0.073601	0.104831
125m	0.085993	0.008456	0.063806	0.086946
350m	0.060566	0.005893	0.045165	0.053475
800m	0.031424	0.003277	0.033789	0.037172
2000m	0.010783	0.001955	0.015529	0.030997



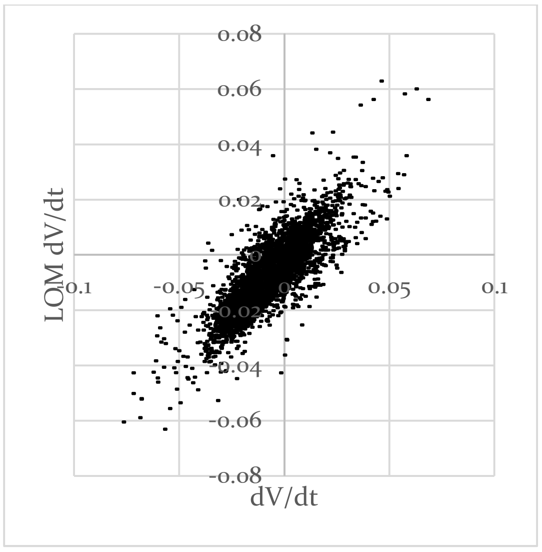


Figure 4-7 LOM water current forecast. On the left, scatterplot of actual change in surface water current for the u (east-west) component vs forecasted change. On the right, the same, except for the v (north-south) component.

4.6 Pruning

Networks were trained for every variable for every level. However, because of the way the networks are designed, a network for a variable for one level can be used interchangeably with a network trained for a different level.

When cross-testing networks across levels, it was found that the network for salinity from level 0 was actually the best performing network for all levels less than or equal to 17. Although all the networks trained for above level 17 performed better than NCOM, the entire model performed better when the lower performing networks were swapped out in favor of the better performing ones that were trained for different levels.

The same was found for levels greater than 17, where the best performing network was from level 28.

4.7 Discussion

LOM is a much less computationally expensive model compared to NCOM. This will allow LOM to compute higher resolution grids than NCOM, and with smaller time-steps. Both of these would improve accuracy over NCOM.

LOM already performs better than NCOM in shallow water with forecasting temperature and salinity. Forecasting water current has been a significantly bigger challenge. However, even if water current forecasting cannot be improved, the results show that a hybrid LOM-NCOM model that focused on each's strengths could result in a much more accurate model.

The strong performance of NCOM is very misleading. Our measurement of the accuracy of NCOM is primarily on the comparison of a 24 hour NCOM forecast to the ground truth initialization grid for NCOM for the same time the next day. According to Allan Wallcraft (Navy Research Lab, NCOM), this is not a fair comparison and leads to a measurement of error that is much lower than the actual forecast error [27]. This is because of the fact that the ground truth initialization grid is actually based on the 24 hour forecast from the day before… exactly what we're trying to find the error in. The initialization values are a weighted average of the 24 hour forecast from the previous day, buoy data, and satellite data. Because there are so few subsurface measurements, the 39 layers below the surface are synthetic profiles based on estimates from surface observations [27]. For these reasons, our error measurement is biased towards a lower error for NCOM.

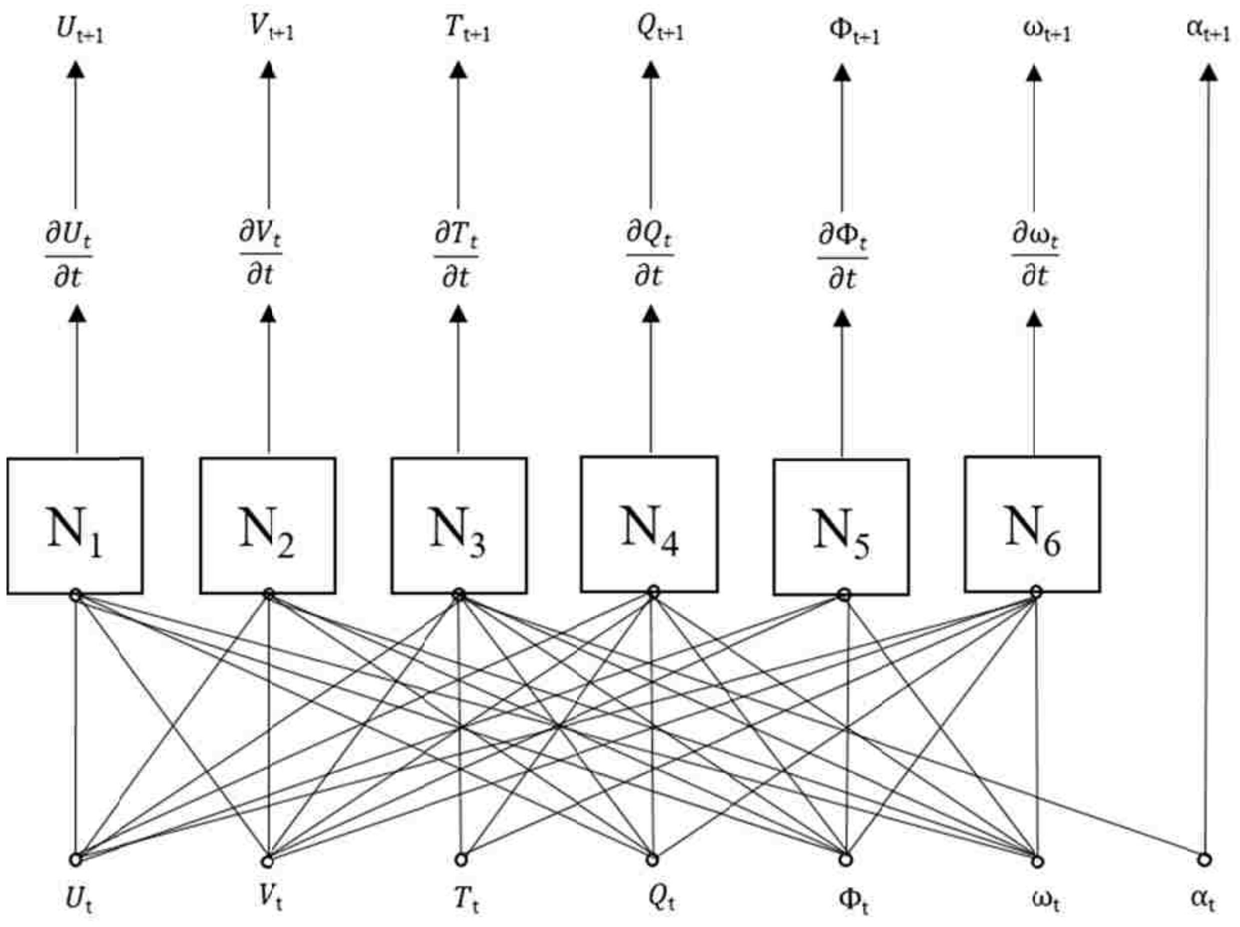
The results show that LOM does have forecast skill and can outperform a state-of-the-art ocean forecast system (NCOM), especially if LOM is further developed or combined with NCOM.

Additional numerical stability could be achieved by switching from the leapfrog time integration scheme to a 3rd order Runge-Kutta integration scheme.

5 CASE STUDY: ATMOSPHERIC MODELING

5.1 INTRODUCTION

This chapter describes an implementation of a mesoscale meteorological model using the proposed methodology. In this instance, we have the current values of all variables at t=0 on a 56x49x36 grid, and the ground truth values 1 hour later.



U_{t+1}
V_{t+1}
T_{t+1}
Q_{t+1}
Φ_{t+1}
ω_{t+1}
α_{t+1}
$\frac{\partial U_t}{\partial t}$
$\frac{\partial V_t}{\partial t}$
$\frac{\partial T_t}{\partial t}$
$\frac{\partial Q_t}{\partial t}$
$\frac{\partial \Phi_t}{\partial t}$
$\frac{\partial \omega_t}{\partial t}$
N_1
N_2
N_3
N_4
N_5
N_6
U_t
V_t
T_t
Q_t
Φ_t
ω_t
α_t

Figure 5-1: Networks N1 through N6 are independent and can be trained separately. Each network computes the time derivative of a different meteorological observable variable.

A complete forecast system would forecast N1 through N6.

Unlike the 3-hourly ground truth data for LOM, 1-hour later ground truth is available.

Figure 5-2 shows two applications of the network N3. The set of networks can be applied recursively to step forward in time any number of times. Not shown is similar networks used to forecast or update U, V, C, and S.

Variables are the u and v components of the wind, temperature, cloud cover, and the sun's altitude angle.

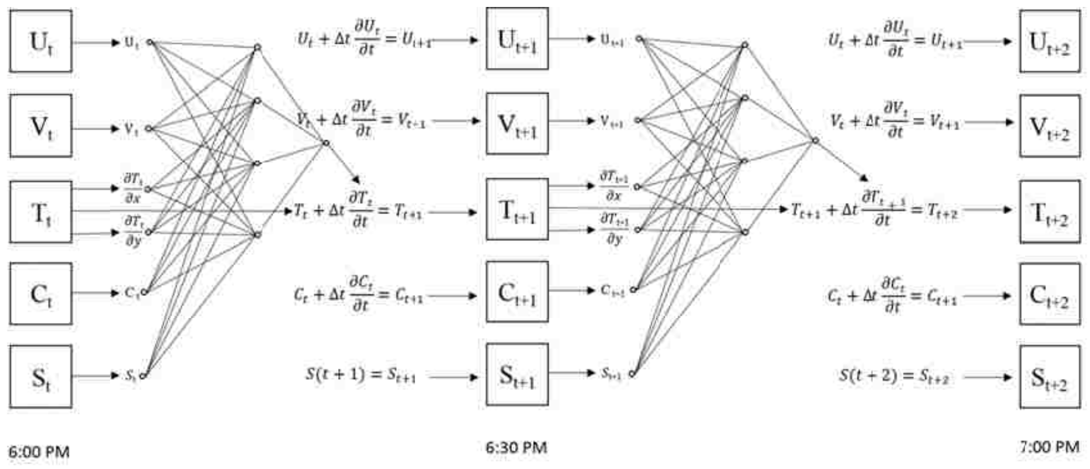

U_t
V_t
T_t
C_t
S_t
$U_t + \Delta t \frac{\partial U_t}{\partial t} = U_{t+1}$
$V_t + \Delta t \frac{\partial V_t}{\partial t} = V_{t+1}$
$T_t + \Delta t \frac{\partial T_t}{\partial t} = T_{t+1}$
$C_t + \Delta t \frac{\partial C_t}{\partial t} = C_{t+1}$
$S(t+1) = S_{t+1}$
U_{t+1}
V_{t+1}
T_{t+1}
C_{t+1}
S_{t+1}
$T_{t+1} + \Delta t \frac{\partial T_{t+1}}{\partial t} = T_{t+2}$
$S(t+2) = S_{t+2}$
U_{t+2}
V_{t+2}
T_{t+2}
C_{t+2}
S_{t+2}
6:00 PM
6:30 PM
7:00 PM

Figure 5-2 Temperature forecasting recurrent network unfolded though time with time step Δt equal to 30 minutes. U and V are the east-west and north-south components of the wind speed, respectively. T is the temperature, C is the cloud cover, and S is the sun's altitude above the horizon. Outputs of the network do not directly re-enter as inputs. Instead, they must go through a pre-processing stage, which complicates backpropagation training.

5.2 LM3: Learned Mesoscale Meteorological Model

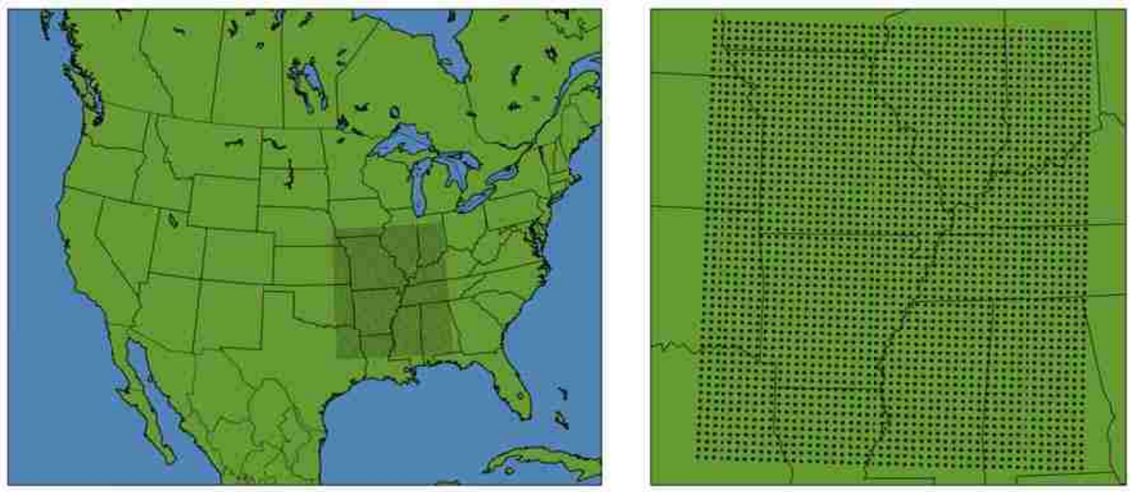


Figure 5-3: NCEP Grid 252, a 301x225x37 grid of observations with approximately 20km horizontal resolution. Only the 56x49 grid over the southeastern US is used. This represents a roughly homogenous region with similar elevation and no ocean

This is the main version of the weather model and the only version that has been implemented.

5.2.1 The grid and the forecast

The input data to forecast wind is the wind speed in the east-west (U) and north-south (V) directions, geopotential height, and latitude at 2744 locations across the southeastern United States. The input data to forecast temperature is temperature, wind, cloud cover, and solar angle. These observations are on a 56x49 grid as shown in fig. 5-3.

The goal of the forecast is to determine the future state of the gridded variables 1 hour in the future. To accomplish this, a 6-minute forecast is generated for every point on the grid. This 6-minute forecast can be further extended by using it as the input to the forecast system again to time step further and further into the future. This is done 10 times to generate a forecast 1 hour in the future.

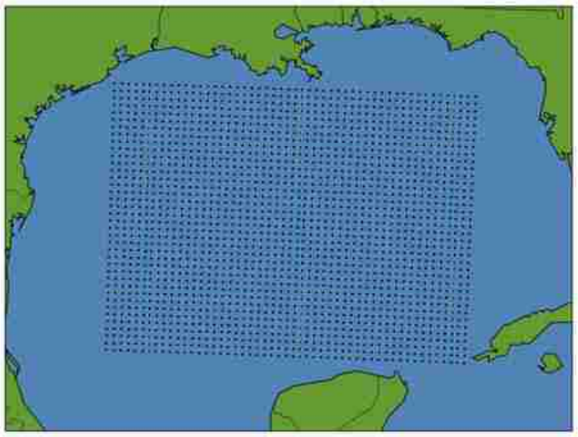
5.2.2 Inputs and outputs

The recurrent networks to forecast U, V, and T each require 5 inputs and generate 1 output. This is possible because of our pre-processing stage where we compute the spatial derivatives at the point we wish to forecast.

Alternately, instead of computing the partial derivatives for use as inputs to the network, we could train an autoencoder network. This could take the 6 nearest neighbor grid points as inputs and use unsupervised learning to learn a lower dimensional representation.

If we instead input every temperature value in the 1-region, every U and V component of wind, then this would be 12 more inputs – a total of 17 inputs to the network. Because training slows and the network becomes less able to capture the desired function with increased dimensionality, this is undesirable.

5.3 HLM3: Hurricane Learned Mesoscale Meteorological Model



This variation has not been implemented yet, but results from simply changing the region to that of the Gulf of Mexico. This region was carefully selected because it’s the largest rectangular subgrid of the NCEP Grid 252. This grid would be nested inside a larger grid. Ideally, this would cover all the water of the Gulf of Mexico and LM3 would handle the land regions.

This model would need to take into account different input variables from LM3, like sea surface temperature and have a smaller time step to accurately handle the higher winds.

5.4 Ensemble Forecasting

The initial state of the atmosphere is not perfectly known. This introduces uncertainty into the forecast. In meteorology, the conventional approach to handle this is to perturb the input data in various ways and rerun the model. If the resulting forecast is similar to the forecast before, then we can be confident in the result. If the resulting forecast is significantly different, then we know the forecast is very sensitive to input errors and we should not be as confident in the resulting forecast.

For a neural network implementation of a dynamical meteorological model, we can accomplish this in an additional way. Not only can we perturb the input data to determine the sensitivity to errors in the input, but we can also train multiple neural networks with different architectures and weights. In this way we can determine the sensitivity to slight variations in model architecture.

The results of all the forecasts can be averaged to produce a more accurate final forecast.

5.5 Implementation and experimental setup

The proposed method was implemented in Python and C++. All neural network code was written by the author specifically for this task. The experiment was run on a laptop with a 1.6GHz Intel Core 2 Duo U7600 processor and 4GB RAM. Training time was limited to 1 day, but could be allowed to run longer for reduced error. Because wind speed, temperature, relative humidity, specific humidity was forecast on 37 levels of the atmosphere, plus geopotential height at the surface, this required training 186 different networks.

5.6 Data

The network was trained using the hourly input data sets to the Rapid Refresh (RAP) model for every third day in January 2014 beginning with the 3rd (3, 6, 9, 12...), and validated against every third day that same month beginning with the 1st (1, 4, 7, 10...).

The RAP model is run hourly out to 18 hours on a 301x225 Lambert conformal projected grid with a 20km horizontal resolution and 37 vertical levels with pressure coordinates from 1000mb to 100mb. Data includes temperature, wind speed/direction, geopotential height, relative humidity, and vertical velocity for every level. This data can be downloaded from either the NCEP or NCDC ftp server [33, 35], or captured via NOAAPORT satellite broadcast.

Figure 5-4: My 10' NOAAPORT dish. Due to large download sizes, going forward input data will be captured from the NOAAPORT satellite broadcast network.

Because the same learned network is applied to every grid point on a given pressure surface to create a forecast, and interaction with land/water at the surface therefore needs to be taken into account, this effect is reduced by selecting a 56x49 sub-grid that covers the southeastern US and no ocean. This is a roughly homogenous region. This is only necessary at the lower levels of the atmosphere that are most influenced by interaction with land. This region of the atmosphere is known as the planetary boundary layer.

Training and validation data was generated by computing input data for every grid point and generating target values for a 1 hour forecast by using the input files for the RAP model initialized 1 hour later.

5.7 Results

5.7.1 Computational Complexity of LM3 vs RAP

LM3 works on a 301 x 225 x 37 grid (or 2,505,825) and takes approximately 6.5 seconds to compute one time-step on a Surface Book laptop.

RAP (13km) works on a 451 x 337 x 37 grid (or 5,623,519 grid points) and takes approximately 16 minutes on 320 Intel Cores [36].

HRRR (3km) takes approximately 41 minutes on 1120 Intel Cores.

This computation is done on a 208 teraflop supercomputer at WCOSS with 10,048 processing core. This means RAP uses 3.18% of WCOSS and HRRR takes approximately 11.15%. These allocations correspond to 6.6 teraflops and 23.2 teraflops, respectively.

This means a run of the RAP model consists of 9,408 trillion floating point operations, and HRRR consists of 84,378 trillion floating point operations. On the LM3 grid, the RAP model would need approximately 4,192 trillion floating point operations.

If LM3 is running on an estimated 100 gigaflop machine for 6.5 seconds, it takes approximately 650 billion floating point operations per time-step. To forecast out to 18 hours with a 4-minute time-step, it would take an estimated 175.5 trillion floating point operations.

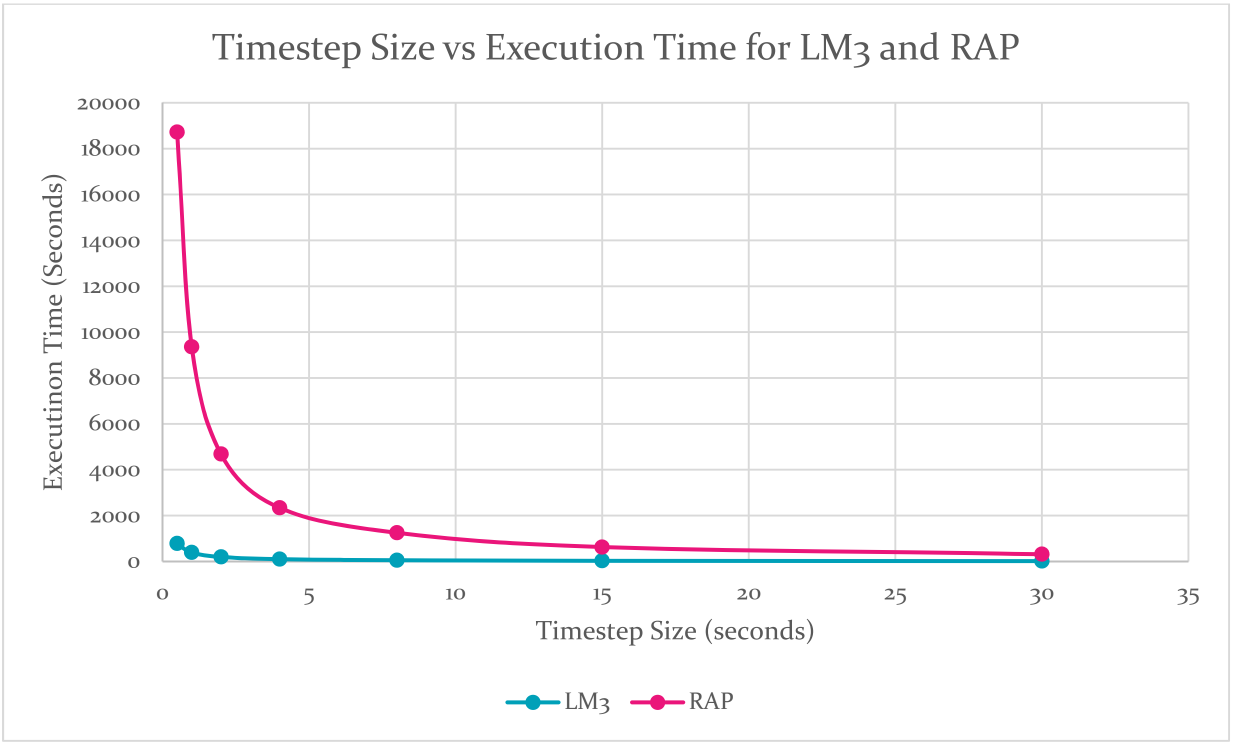


Figure 5-5 Time-step Size vs Execution Time for LM3 and RAP for a 1 hour forecast

Figure 5-5 shows how the execution time of both LM3 and RAP vary with time-step size for a 1 hour forecast. The smaller the time-step, the more pronounced the difference in execution time becomes. For time-steps under 5 minutes, the effect is magnified significantly because halving the time-step results in double the number of iterations required to create the same 1 hour forecast.

Under these assumptions, LM3 would be approximately 16 times faster than RAP.

With a significant speedup over RAP and HRRR available, this would allow LM3 to work on a much higher resolution grid and still require fewer computational resources. The higher resolution grid would allow LM3 to resolve smaller features, which would help offset any potential reductions in accuracy.

5.7.2 LM3 Wind Forecasting Results

Table 5-1 Summary of results forecasting U for 10 levels of the atmosphere, where LM3 represents the results of the proposed method. Error is MAE in m/s (meters per second).

Level	LM3 U	RAP U	Persistence U	Climate U
1000	0.4514	0.4368	0.8707	2.2352
900	1.0420	1.0441	1.8615	2.4850
800	0.9070	0.9642	1.5217	3.7477
700	1.0209	0.7399	1.7444	5.2456
600	1.1019	0.7201	1.5368	6.7507
500	1.0215	0.5598	1.1289	7.6454
400	1.1763	0.7498	1.6150	7.8614
300	1.4445	1.0059	2.4442	9.2635
200	1.4522	1.4724	1.6286	6.4824
100	0.7468	0.5429	1.3671	3.9258

LM3 was more accurate than RAP at forecasting the u-component of wind speed (east-west) on the 900 mb, 800 mb, 825mb, 200 mb, 175mb, 150mb, and 125 mb levels.

Table 5-2 Layers where LM3 is more accurate than RAP at forecasting the u-component of the wind

Level	LM3 U Error	RAP U Error
900	1.0420	1.0441
825	0.9754	1.0144
800	0.9070	0.9642
200	1.4522	1.4724
175	1.1297	1.3357
150	0.8886	1.2160
125	0.8044	0.8527

Table 5-3 Summary of results forecasting V for 10 levels of the atmosphere, where LM3 represents the results of the proposed method. Error is MAE in m/s (meters per second).

Level	LM3 V	RAP V	Persistence V	Climate V
1000	0.5807	0.7011	1.2171	3.4181
900	1.0868	0.8296	2.0038	10.5488
800	1.0764	0.9353	1.7772	9.7302
700	1.0714	0.7266	2.3987	8.2690
600	1.2007	0.5436	2.0687	6.8842
500	1.1961	0.8981	2.3799	6.2477
400	1.4151	0.9954	1.6528	6.7325
300	2.4708	1.2572	2.7266	7.9448
200	1.6046	1.0807	2.7472	5.1649
100	1.0091	0.6839	1.6632	3.7913

LM3 was more accurate than RAP at forecasting the v-component of wind speed (north-south) at the surface (1000 mb).

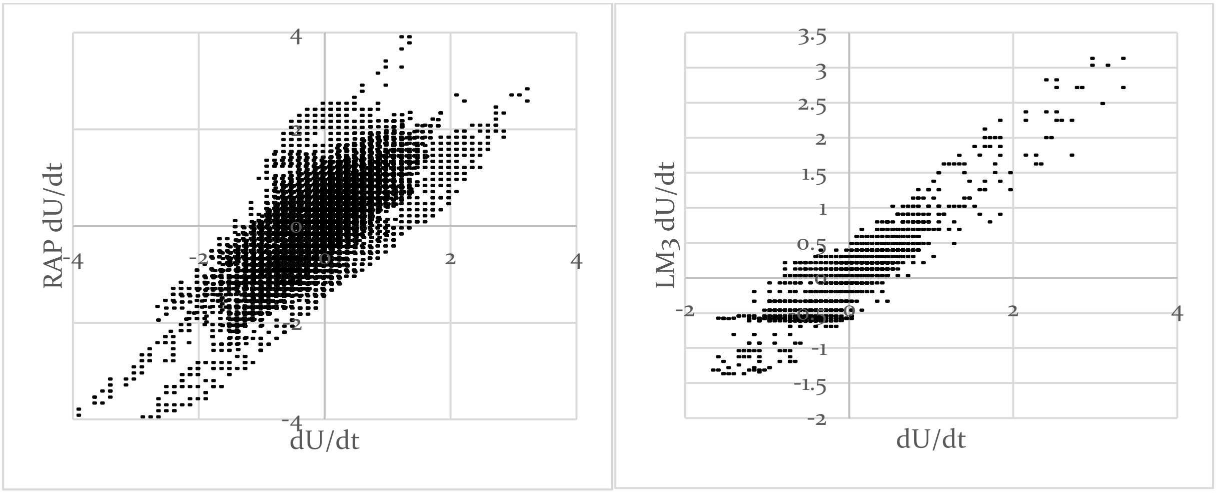


Figure 5-6: Scatterplot for U Forecast on 1000mb Level

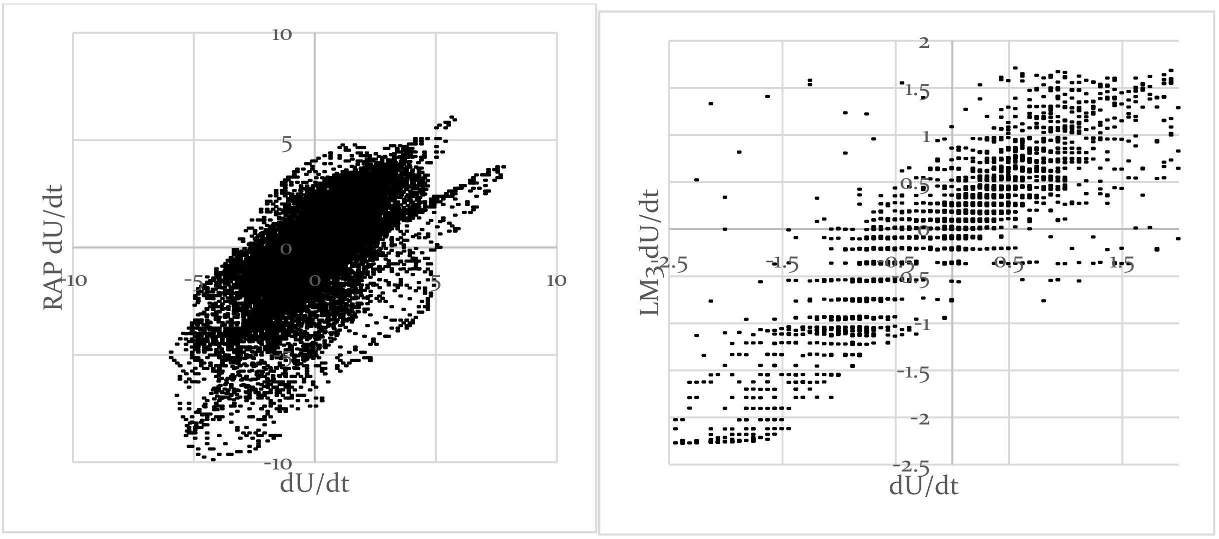


Figure 5-7: Scatterplot for U Forecast on 500mb Level

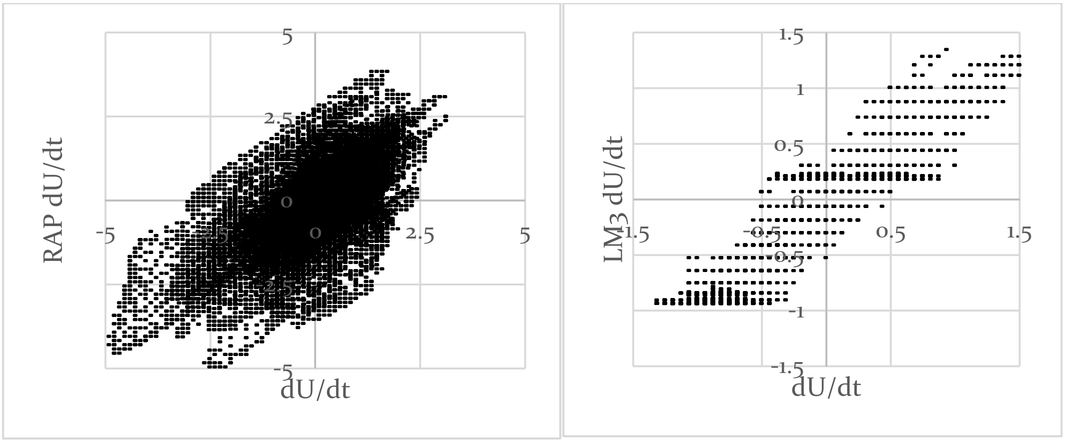


Figure 5-8: Scatterplot for U Forecast on 100mb Level

5.7.3 LM3 Temperature Results

Table 5-4 Summary of results forecasting temperature for 10 levels of the atmosphere, where LM3 represents the results of the proposed method. Error is MAE in Celsius.

Level	LM3 Temp Error	RAP Temp Error	Persistence Temp Error	Climate Temp Error
1000	0.9631	0.6820	1.7676	6.4362
900	0.4248	0.2486	0.6389	8.0903
800	0.3941	0.2285	0.5309	6.8417
700	0.3080	0.2545	0.5322	4.9478
600	0.2612	0.1825	0.3302	2.5596
500	0.2820	0.2429	0.4123	1.1399
400	0.3134	0.2445	0.4932	1.2228
300	0.3896	0.2641	0.6745	1.4285
200	0.7865	0.3393	0.8251	2.2704
100	0.3101	0.3718	0.5015	3.9936

As can be seen in Table 5-5, LM3 was more accurate than RAP at forecasting the temperature on the top 3 levels of the atmosphere (100mb, 125mb, and 150mb), and competitive elsewhere. LM3 consistently beats the persistence and climatology models.

Table 5-5 Layers where LM3 is more accurate than RAP for forecasting temperature

Level	LM3 Temp Error	RAP Temp Error
150	0.27545	0.37862
125	0.26573	0.28615
100	0.31015	0.37188

5.7.4 LM3 Relative Humidity Results

Table 5-6 Summary of results for forecasting relative humidity for levels that are multiples of 100

Level	LM3 RH Error	RAP RH Error	Persistence RH Error	Climate RH Error
1000	3.43831	4.244534	7.754373	13.60183
900	4.748244	5.136297	6.485058	26.52671
800	4.535183	2.720845	7.117711	28.12656
700	5.289009	3.96793	8.984694	27.59054
600	5.310895	3.281706	8.729956	28.20039
500	6.171344	4.654155	12.73105	24.61318
400	7.335474	3.704082	14.27697	19.19148
300	6.120434	4.541181	11.84767	14.89603
200	1.910719	0.764213	2.873907	5.688638
100	0.197648	0.206997	0.3207	2.083975

LM3 was more accurate than RAP for forecasting relative humidity on the first 5 layers closest to the ground (1000mb – 900mb), plus the 100mb level at the top of the atmosphere. In addition, LM3 consistently beat the persistence and climatology models.

Table 5-7 Layers where LM3 is more accurate than RAP for forecasting relative humidity

Level	LM3 RH Error	RAP RH Error
1000	3.43830	4.2445
975	3.35697	3.83928
950	3.47262	3.83928
925	4.24169	4.90451
900	4.90451	5.13629
100	0.19764	0.20699

5.7.5 LM3 Specific Humidity Results

Table 5-8 Summary of results for forecasting specific humidity for levels that are multiples of 100

Level	LM3 U	RAP U	Persistence U	Climate U
1000	0.00022	0.00038	0.000464	0.001622
900	0.000399	0.000472	0.000543	0.001612
800	0.000296	0.000177	0.000472	0.00156
700	0.000268	0.000188	0.000436	0.001173
600	0.000167	9.43E-05	0.000277	0.000873
500	0.000104	6.92E-05	0.000204	0.000413
400	5.38E-05	2.26E-05	9.66E-05	0.000134
300	1.34E-05	7.44E-06	2.32E-05	3.34E-05
200	3.10E-06	1.32E-06	4.20E-06	8.84E-06
100	1.66E-07	2.03E-07	2.53E-07	8.03E-07

LM3 was more accurate than RAP on the first 6 layers nearest to the ground for forecasting specific humidity (1000mb through 875mb), plus the 100mb level at the top of the atmosphere.

Table 5-9 Layers where LM3 is more accurate than RAP for forecasting specific humidity

Level	LM3 SH Error	RAP SH Error
1000	0.00022022	0.0003803
975	0.00025526	0.0003979
950	0.00039793	0.0005065
925	0.00036531	0.0005084
900	0.00039891	0.0004716
875	0.00040037	0.0004017
100	1.6635e-07	2.0277e-07

5.8 Pruning

Networks were trained for every variable for every level. However, because of the way the networks are designed, a network for a variable for one level can be used interchangeably with a network trained for a different level.

When cross-testing networks across levels, it was found that some networks outperformed all others from other layers, even though they were only trained on data from their own layer. For instance, for forecasting relative humidity, the best network for the top 12 layers of the atmosphere was the network trained only with data from the top 1 layer. The middle portion of the atmosphere was best forecast by a network that was trained on data only from layer

Table 5-10 Table of variables and layers, and the network that best forecasts that variable.

Pressure	Layer #	T best	U best	V best	RH best	SH best
100	36	36	36	36	36	32
125	35	35	35	35	36	32
150	34	34	34	34	36	32
175	33	33	33	33	36	33
200	32	32	32	30	36	32
225	31	31	12	31	36	32
250	30	30	30	24	36	26
275	29	30	29	17	36	26
300	28	30	28	28	36	26
325	27	27	27	27	2	26
350	26	26	26	26	36	26
375	25	25	25	25	36	26
400	24	24	0	24	21	24
425	23	24	0	23	21	23
450	22	22	21	22	21	22
475	21	20	21	21	21	22
500	20	18	20	20	21	21
525	19	18	19	19	21	21
550	18	18	18	18	36	21
575	17	17	17	17	21	12
600	16	17	16	17	21	15
625	15	13	14	15	21	13
650	14	13	14	14	21	13
675	13	13	13	14	21	12
700	12	12	12	12	21	12
725	11	10	10	11	21	11
750	10	10	10	10	21	11
775	9	10	9	9	36	21
800	8	10	8	8	36	21
825	7	18	7	6	2	21
850	6	18	6	6	2	26
875	5	10	5	5	2	3
900	4	18	4	4	36	18
925	3	18	3	3	36	11
950	2	18	0	2	2	11
975	1	23	1	1	7	18
1000	0	18	0	0	7	11

21, at 450mb… very near the center of mass of the atmosphere. The lower portion was best forecast by networks from layer 2 and layer 7, as can be seen in Table 5-10.

Similar patterns were found in networks trained for other variables.

5.9 Training Error vs Epoch

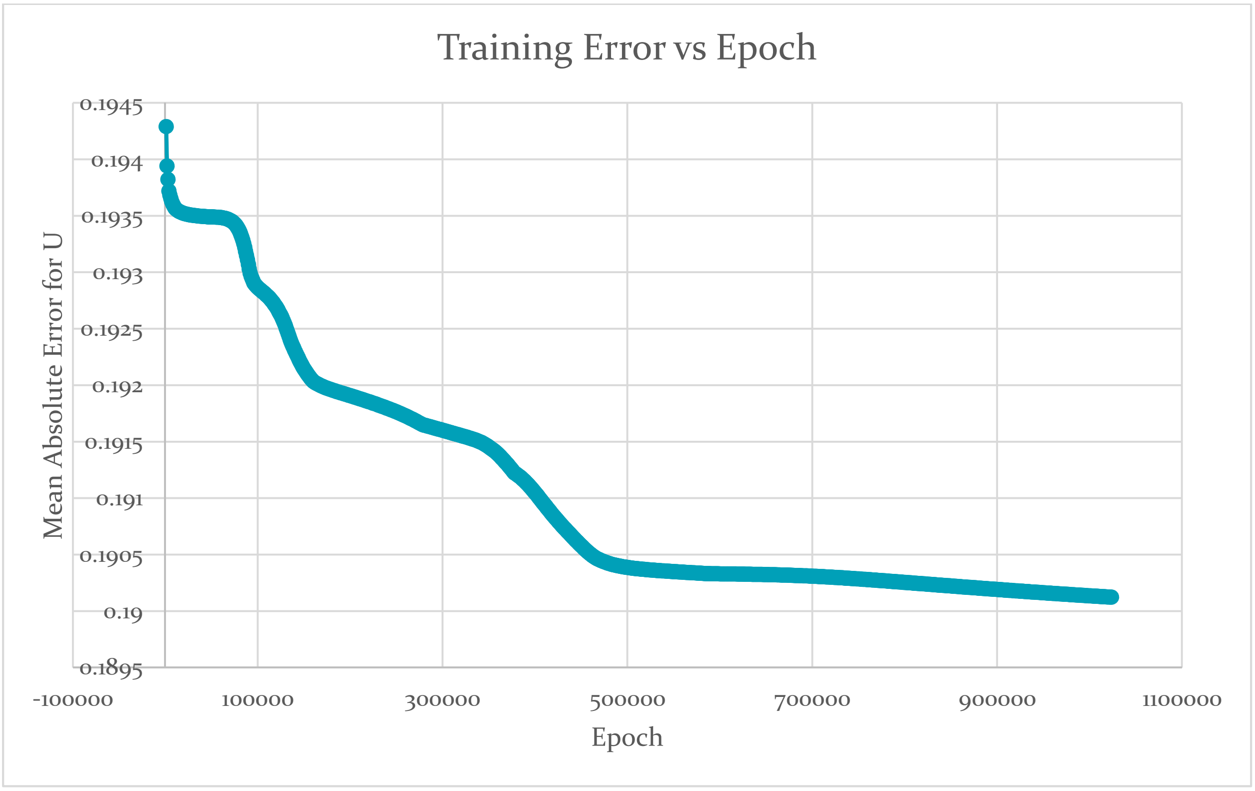


Figure 5-9 Training error vs epoch.

Figure 5-9 shows training error versus epoch for the first 1,024,000 iterations of training for the 1000 mb level u-component forecasting network for 1 hour.

5.10 Discussion

The RAP model is an operational model run hourly by NCEP, and represents typical results by a sophisticated primitive equation model.

Similarly as discussed before in Section 4.7 about the NCOM model, the strong performance of RAP is equally misleading. Our measurement of the accuracy of RAP is primarily on the comparison of a 1-hour RAP forecast to the ground truth initialization grid for RAP for the same time 1 hour later. As was the case with NCOM, the ground truth initialization grid is actually a weighted average of the 1-hour forecast from 1 hour ago plus any new observations. Most weather measurements are ground-based, and even satellite based observations don't provide a 3D insight into the atmosphere. The only data available for above ground level is radiosonde data on weather balloons, and those are only released twice daily at 0 GMT and 12 GMT. For these reasons, there is little to weight the previous 1-hour forecast away from itself. Therefore, this measure of error is likely biased towards lower measures of error for the RAP model.

The implementation of the proposed method only forecasts horizontal wind speed, temperature, relative humidity, and specific humidity. Vertical wind speed and geopotential height are calculated for each level using diagnostic equations. A complete model would forecast all model variables. Many other important components are not yet implemented and are assumed to remain static during the forecast, but are necessary for more accurate, competitive results. Despite these limitations, Figures 5-5, 5-6, and 5-7 show that the learned behavior closely mirrors the desired behavior.

Figure 5-5 is a scatterplot comparing the actual to the forecasted one-hour change in wind speed for both the proposed method and the RAP model for the 1000mb level. The 1000mb level closely follows the surface at ground level. The proposed method outperforms the RAP model on the 125mb - 200mb, 800 - 825mb, and 900mb levels, as can be seen in Table 5-2 and the 1000mb level as can be seen in Table 5-3. LM3 also outperforms the RAP model in forecasting temperature in the top 3 layers of the atmosphere, as can be seen in Table 5-5, in forecasting relative humidity in 6 layers, as can be seen in Table 5-7, and in forecasting specific humidity, as can be seen in Table 5-9.

Figure 5-6 is the same as Figure 5-5, except for the 500mb level. This level is shown because it represents the approximate center of mass of the atmosphere. It performs slightly worse than the RAP model in terms of MAE, but the scatterplot shows it more closely follows the line Y=X.

Figure 5-7 is the same as Figure 5-6, except for the 100mb level. This level represents the top of the atmosphere.

In the planetary boundary layer at the surface, learned networks should only be shared with regions with similar surface characteristics, like albedo, elevation, and land use type. Because of the homogenous nature of the boundary layer over water, this approach could be particularly well-suited to forecasting over oceans and could be applied to forecasting tropical systems like hurricanes and typhoons out at sea. However, special consideration would have to be made for landfalling systems and an appropriate time step would have to be chosen that satisfies the CFL condition.

Additional numerical stability could be achieved by switching from the leapfrog time integration scheme to a 3rd order Runge-Kutta integration scheme, which is used by the RAP model

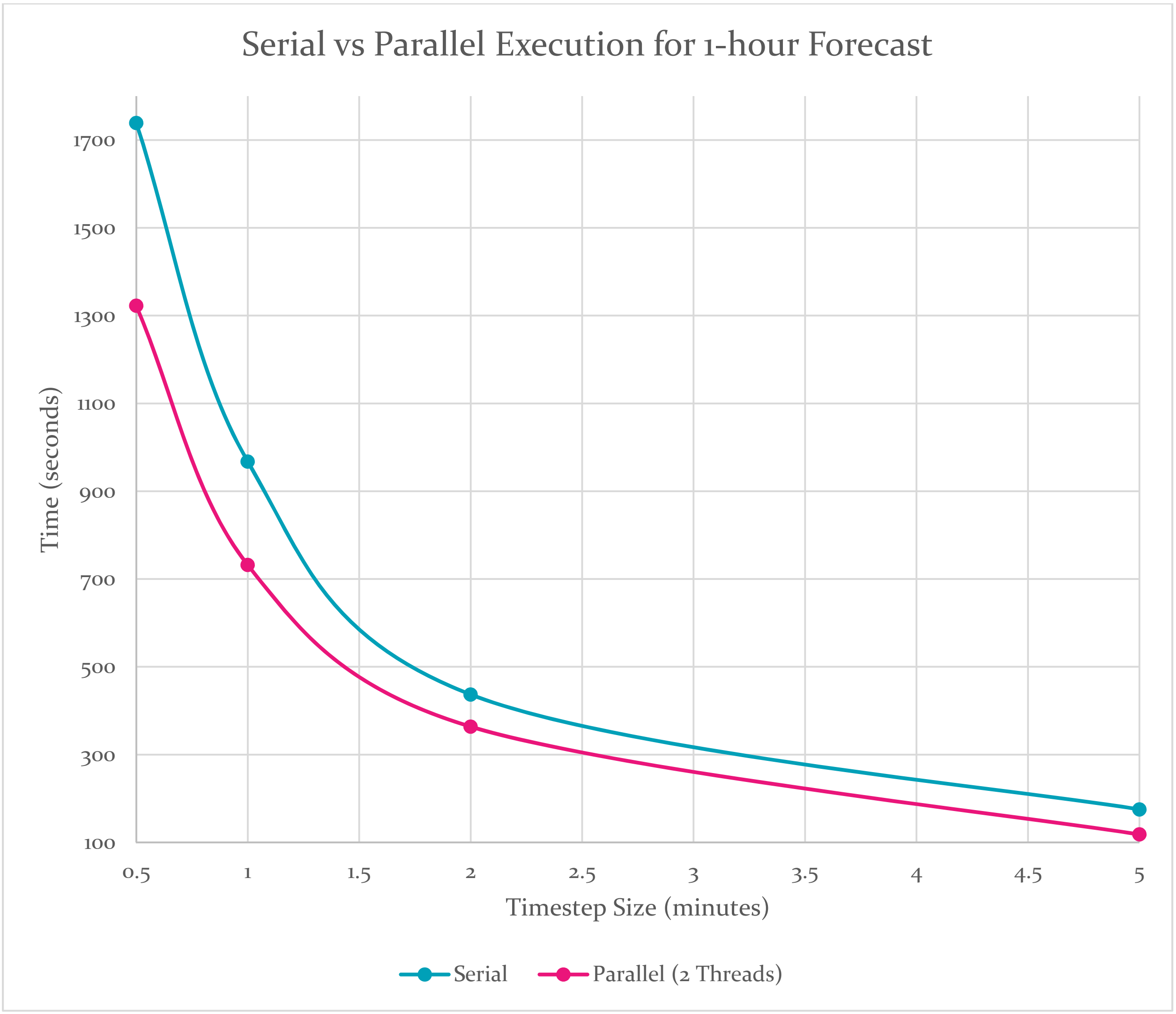
[30]. The time step, 6 minutes in our implementation, could also be brought down to 1 minute to match the RAP model.

6 Results of Parallelization

6.1 Introduction

In this chapter, we look at the speedup achieve when parallelizing the code. Parallelism is introduced using OpenMP. The system could be further parallelized using MPI to spread computation across machines.

6.2 Results



Parallelizing the implementation of LM3 resulted in an average speed up of approximately 33% over the serial implementation. Running time could be significantly reduced with further parallelization.

Execution times increase significantly with smaller time-step. Reducing the time-step by half results in twice as many iterations required to create a forecast of the same length.

7 Conclusion

7.1 Summary

The proposed RNN-based forecast model can be used to create a fully learned ocean or weather model. In order to do this, further work must be done to forecast all variables, for all regions and land types. Special networks need to be trained to forecast over oceans, in the mountains, in forested regions, and over cities, although these specialized networks are only required for the lower levels of the atmosphere or the ocean.

Both LOM and LM3 showed forecasting skill by outperforming both the persistence and climatology models.

Even with the limited implementation, the implementation of the proposed approach outperformed the RAP model at forecasting wind speed in the north-south direction on the 1000mb level – the level nearest the ground, the 125 mb, 150 mb, 175 mb, 200 mb, 800 mb, and 825 mb levels for wind speed in the east-west direction, temperature at the 100 mb, 125 mb, and 150 mb levels, specific humidity on the first 6 layers above ground level (1000 mb – 875 mb) and the 100 mb level, relative humidity on the first 5 layers above ground level (1000 mb – 900 mb) and the 100 mb level, and performed competitively elsewhere.

Even more encouragingly, LOM outperformed NCOM in salinity forecasting for layers 1 through 16 and temperature forecasting for layers 2 through 16, showing significant skill in forecasting temperature and salinity in shallow water.

In addition to these good forecasting results, the LOM model produced a forecast 3.6 times faster than NCOM, and the LM3 model produced a forecast 24 times faster than RAP. These efficiencies can be used to increase the resolution and time-step the model uses to further increase accuracy while still requiring the same or fewer computational resources as RAP and NCOM.

For improved weather forecasting, networks also need to be designed to forecast phase change and latent heat. The remaining work there mainly involves selecting the appropriate inputs to consider. With this, we would have a full forecast system.

7.2 Benefits of the Proposed Approach

The proposed method provides numerous advantages over prior neural network approaches. First, learning is resolution independent. It is independent of both the resolution of the input data and the resolution it is trained at. This is achieved by utilizing finite differences as inputs and the time derivative output.

Second, learning is model independent. No forecast model data is used in the forecast process. Input data is the analysis data and can be replaced by any objective analysis routine to produce gridded input data. This flexibility means that we can make changes without retraining the network.

Third, the method is more efficient. Here, we are effectively replacing a complicated set of non-linear partial differential equations with a neural network, which are commonly used as universal function approximators. As we have discussed in previous sections, NCOM is 3.6 times more computationally expensive as LOM, and RAP is nearly 24 times more computationally expensive as LM3.

Forth, the proposed approach works for an entire region instead of only a single location. There have been many proposals for weather forecasting neural networks that only look at the time series data for a single location, but completely ignore the vital spatial component. For this reason, prior approaches are incapable of forecasting the movement of cyclones and frontal boundaries.

Fifth, the method can learn from its mistakes and improve without human intervention. Parameterization schemes are approximating equations used to estimate complicated processes and sub-grid scale phenomena, like 3km diameter fair weather cumulus clouds on a 10-20km grid. The proposed approach seeks to learn the governing equations instead of employing human-created estimations.

Finally, the proposed approach is applicable to learning to forecast many things more than just the ocean and weather. Differentiation-Integration Time Step networks should be applicable to many more time series prediction problems.

8 Further Study

Although the LOM ocean model is already more efficient than the Navy NCOM model, the efficiency of the LOM can still be improved significantly. One of the largest computations in LOM is copying data from a row-column format to a linear one... and back. This overhead of unnecessarily moving millions of values every time-step could be eliminated by re-writing the neural network library to directly accept the input as-is.

The initial work on the LM3 meteorological model only seeks to forecast a limited subset of variables. However, for best results we must consider the evolution of all relevant variables. As currently implemented, our model assumes the other variables are static. This is the largest contributor to the error. Further work would be directed towards forecasting these other important variables.

Training LM3 using HRRR 3km data would be extremely helpful because this model provides intermediate output every 15 minutes, instead of the 1-hour RAP data used currently. Unfortunately, only the 1-hour HRRR data is publicly available at this time.

Finally, further experiments with parallelization could be performed to further speedup training and forecast times. Although the current implementation runs well on a laptop computer, creating forecasts for larger regions would greatly benefit from spreading computations across multiple machines.

References

[1] D. M. Fratantoni, A. Gellers and N. Sharma, "On The Oceanography of Brazil's Equatorial Magin: Hazardous Offshore Currents and Strategies for Mitigation," in *Rio Oil & Gas Expo and Conference*, Rio de Janeiro, 2014.

[2] Horizon Marine, Inc, *Eddy Lazarus & the Loop Current – Paralyzing the Gulf in 2014,* American Association of Drilling Engineers, 2015.

[3] R. Aleimda, "Major Downtime for Gulf of Mexico Energy Majors as Loop Current Halts Operations," 27 October 2014. [Online]. Available: http://gcaptain.com/major-downtime-gulf-mexico-energy-majors-loop-current-halts-operations/.

[4] W. McCulloch and W. Pitts, "A Logical Calculus Of The Ideas Immanent In Nervous Activity," *Bulletin of Mathematical Biophysics,* vol. 5, pp. 115-133, 1943.

[5] T. M. Mitchell, Machine Learning, Singapore: McGraw-Hill, 1997.

[6] S. Marsland, Machine Learning: An Algorithmic Approach, Boca Raton: CRC Press, 2009.

[7] G. E. Hinton, "Learning multiple layers of representation," *Trends in Cognitive Sciences,* vol. 11, no. 10, pp. 428-434, 2007.

[8] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations In The Microstructure Of Cognition*, vol. 1, Cambridge, MIT Press, 1986, p. 336.

[9] K. Fukushima, "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics,* vol. 36, pp. 193-202, 1980.

[10] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[11] S. Behnke, "Hierarchical Neural Networks for Image Interpretation," *Lecture Notes in Computer Science,* vol. 2766, 2003.

[12] V. Nair and G. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *International Conference on Machine Learning*, Haifa, 2010.

[13] M. Zakerinia and S. F. Ghaderi, "Short Term Wind Power Forecasting Using Time Series Neural Networks," in *Emerging M&S Applications in Industry and Academia Symposium*, Tehran, 2011.

[14] D. Corne, A. Reynolds, S. Galloway, E. Owens and A. Peacock, "Short term wind speed forecasting with evolved neural networks," in *15th Genetic and evolutionary computation conferenc*, New York, 2013.

[15] R. E. Abdel-Aal, "Hourly temperature forecasting using abductive networks," in *Engineering Applications of Artificial Intelligence*, 2004.

[16] K. G. Abistado, C. Arellano and E. Maravillas, "Weather Forecasting Using Artificial Neural Network And Bayesian Network," *Journal of Advanced Computational Intelligence and Intelligent Informatics,* vol. 18, no. 5, pp. 812-817, 2014.

[17] J. Mao, X. Zhang and J. Li, "Wind Power Forecasting Based On BP Neural Network," in *International Conference On Systems Engineering and Modeling*, Beijing, 2013.

[18] I. El-Feghi, Z. Zubia and S. Abozgaya, "Efficient Weather Forecasting using Artificial Neural Network as Function Approximator," *International Journal of Neural Networks and Advanced Applications,* vol. 1, pp. 49-55, 2014.

[19] K. Raza and V. Jothiprakash, "Multi-Output ANN Model for Prediction of Seven Meteorological Parameters In A Weather Station," *Journal of The Institution of Engineers,* vol. 95, no. 4, pp. 221-229, 2014.

[20] M. Hayati and Z. Mohebi, "Temperature Forecasting Based on Neural Network Approach," *World Applied Sciences Journal,* vol. 2, no. 6, pp. 613-620, 2007.

[21] S. Nurcahyo and F. Nhita, "Rainfall Prediction in Kemayoran Jakarta using Hybrid Genetic Algorithm (GA) and Partially Connected Feedforward Neural Network (PCFNN)," in *Information and Communication Technology*, Bandung, 2014.

[22] S. Baboo and I. K. Shereef, "An Efficient Weather Forecasting System Using Artificial Neural Network," *International Journal of Environmental Science and Development,* vol. 1, no. 4, 2010.

[23] W. Collins and P. Tissot, "Use Of An Artificial Neural Network To Forecast Thunderstorm Location," in *Conference on Artificial Intelligence Applications to Environmental Science*, 2008.

[24] S. Barnes, "A Technique for Maximizing Details in Numerical Weather Map Analysis," *Journal of Applied Meteorology,* vol. 3, no. 4, pp. 396-409, 1964.

[25] G. P. Cressman, "An Operational Objective Analysis System," *Monthly Weather Review,* vol. 87, no. 10, 1959.

[26] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering,* pp. 35-45, 1960.

[27] A. Wallcraft, Interviewee, *NCOM American Seas computation time.* [Interview]. February 2015.

[28] L. F. Richardson, Weather Prediction By Numerical Process, Cambridge University Press, 1922.

[29] T. T. Warner, Numerical Weather and Climate Prediction, New York: Cambridge University Press, 2011, pp. 456-459.

[30] J. Coiffier, Fundamentals of Numerical Weather Prediction, New York: Cambridge University Press, 2012.

[31] D. J. Stensrud, Parameterization Schemes: Keys to Understanding Numerical Weather Prediction Models, New York, Cambridge University Press: Cambridge University Press, 2009, pp. 7-9.

[32] V. M. Krasnopolsky, S. F. Michael and V. C. Dmitry, "New Approach to Calculation of Atmospheric Model Physics: Accurate and Fast Neural Network Emulation of Longwave Radiation in a Climate Model," *Monthly Weather Review,* vol. 133, no. 5, pp. 1370-1383, 2005.

[33] National Centers for Environmental Prediction, "RAP Model Data," [Online]. Available: ftp://ftp.ncep.noaa.gov/pub/data/nccf/com/rap/prod/.

[34] N. D. Walker, R. R. Leben and S. Balasubramanian, "Hurricane-forced upwelling and chlorophyll a enhancement within cold-core cyclones in the Gulf of Mexico," *Geophysical Research Letters,* vol. 32, no. 18, 2005.

[35] National Climatic Data Center, "RAP Model Data," [Online]. Available: http://nomads.ncdc.noaa.gov/thredds/dodsC/rap252/.

[36] B. Kyger, Interviewee, *HRRR Computing Power.* [Interview]. June 2015.

[37] N. Do, K. Udo and A. Mano, "Downscaling Global Weather Forecast Outputs Using ANN for Flood Prediction," *Journal of Applied Mathematics,* vol. 2011, 2011.

[38] I. Maqsood, M. R. Khan and A. Abraham, "Weather Forecasting Models Using Ensembles of Neural Networks," *Neural Computing & Applications,* vol. 13, pp. 112-122, 2004.

[39] R. J. Firth and J. Chen, "Neural Network Implementation of a Mesoscale Meteorological Model," in *International Symposium on Methodologies for Intelligent Systems (ISMIS)*, Roskilde, 2014.

[40] NOAA Earth System Research Laboratory, "Rapid Refresh (RAP)," [Online]. Available: http://rapidrefresh.noaa.gov/.

[41] F. Carr, Interviewee, *Calculating vertical velocity and geopotental.* [Interview]. November 2013.

[42] G. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *AFIPS Conference Proceedings*, 1967.

APPENDIX

SPRINGER LICENSE
TERMS AND CONDITIONS

Apr 07, 2016

This is a License Agreement between Robert Firth ("You") and Springer ("Springer") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Springer, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	3843900907510
License date	Apr 07, 2016
Licensed content publisher	Springer
Licensed content publication	Springer eBook
Licensed content title	Neural Network Implementation of a Mesoscale Meteorological Model
Licensed content author	Robert Firth
Licensed content date	Jan 1, 2014
Type of Use	Thesis/Dissertation
Portion	Full text
Number of copies	5
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	None
Title of your thesis / dissertation	A NOVEL RECURRENT CONVOLUTIONAL NEURAL NETWORK FOR OCEAN AND WEATHER FORECASTING
Expected completion date	May 2016

Estimated size (pages)	60
Total	0.00 USD

Vita

Robert Firth, a native of Santa Rosa, California, received his bachelor's degree in Computer Science from Louisiana State University in 2010. His research interests include machine learning, artificial intelligence, high performance computing, and quantum computing.