# Neural Network Implementation of a Mesoscale Meteorological Model

Robert Firth and Jianhua Chen

Division of Computer Science and Engineering
School of Electrical Engineering and Computer Science
Louisiana State University
Baton Rouge, LA 70808
rfirth1@tigers.lsu.edu,
jianhua@csc.lsu.edu

**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. This paper outlines an alternative approach to developing a mesoscale meteorological model – a modified recurrent 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 in this paper is an extended form of Back-propagation Through Time for a recurrent network with outputs that feed back through as inputs only after undergoing a fixed transformation.

**Keywords:** Recurrent neural networks, spatial-temporal, weather prediction, forecasting, temperature, wind.

## 1 Introduction

While huge accuracy gains have been made in weather forecasting, it still remains a challenging task. Many approaches have been developed including heuristics like persistence and trends, numerical weather prediction, and neural networks.

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 (PDEs), but also the creation of a parameterization scheme to estimate sub-grid scale phenomenon [1].

This paper proposes a method to replace the primitive equations, the set of PDEs that govern atmospheric dynamics [2], with a set of recurrent neural networks.

## 2 Previous Work

Computational approaches that have been developed to forecast weather include heuristics like persistence and trends, numerical weather prediction, and neural networks.

T. Andreasen et al. (Eds.): ISMIS 2014, LNAI 8502, pp. 164–173, 2014.
© Springer International Publishing Switzerland 2014

Numerical weather prediction techniques focus on simulating the evolution of the atmosphere through numerical solutions to the simplified set of partial differential equations known as the primitive equations to find the time derivative of the variables and then applying forward time integration [2]. Models such as the NCEP's RAP (Rapid Refresh) model use this technique to generate forecasts.

Parameterization schemes are used to estimate sub-grid scales phenomena that can't be directly simulated [1]. Krasnopolsky et al. replaced the shortwave and long-wave 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 50-60 times faster computation of the radiation parameterization [6].

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 [3]. 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 [7].

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 [4].

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 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 some domain knowledge.

The learning method presented in this paper 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.

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

## 3 Method

### 3.1 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. 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.

### 3.2 Finite Differences

Many inputs are not fed directly into the network. Instead, they are fed in as partial derivatives with respect to east-west and north-south grid coordinates x and y. These partial derivatives are computed numerically using a centered finite difference scheme:

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

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

### 3.3 Time Integration

The learning task of our recurrent neural network is to learn to compute the partial derivative of each meteorological 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:

$$T_1 = T_0 + \Delta t \frac{\partial T_0}{\partial t}$$

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

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 [10]. It was confirmed experimentally 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.

This same process is used to forecast all meteorological variables.

### 3.4 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 [2].

$$C = \frac{u_{max}\Delta t}{\Delta x} \leq C_{max}$$

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.

### 3.5 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.

This paper describes how to represent and learn N1, N2 (by symmetry with N1), and N3.

### 3.6 Error and Error Attribution

Error is computed by comparing the output of the system to the 1 hour later initialization data for the RAP model.

However, because the network is not a normal recurrent network – the inputs do not directly connect to the inputs – backwards propagation of error is not straightforward. Two rules are therefore created to propagate error back to the output layer of the network (so that normal backpropagation can begin). First, the last step, time integration:

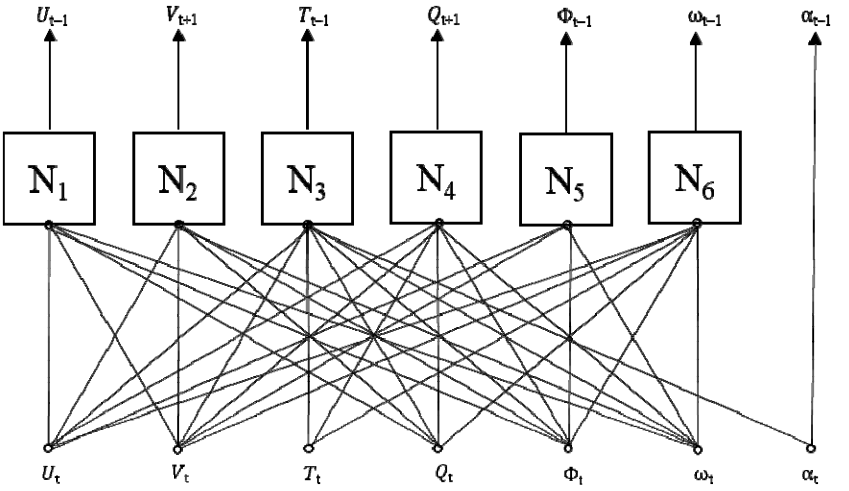


**Fig. 1.** Networks N1 through N6 are independent and can be trained separately. Each network computes the time derivative of a different meteorological observable variable.

$$T_{t+1} = T_t + \Delta t \frac{\partial T_t}{\partial t}$$

Error in $T_{t+1}$ can be attributed to two sources, $T_t$ and $\frac{\partial T_t}{\partial t}$. We can weight this error as:

$$E = (1 - \lambda)E + \lambda E$$

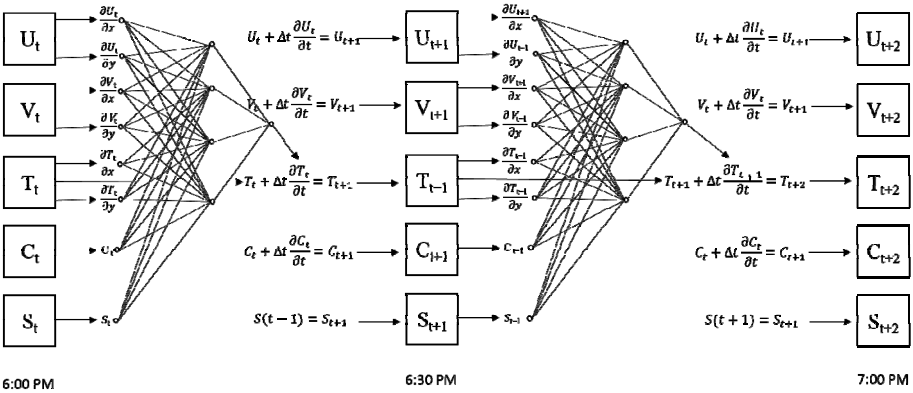


**Fig. 2.** Temperature forecasting recurrent network unfolded though time with time step Δt equal to 30 minutes. U and V are the east-west and north-south components of the wind speed, respectively. T is the temperature, C is the cloud cover, and S is the sun's altitude above the horizon. Outputs of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

Here E is the target – predicted, (1-λ)E is the error attributed to $T_1$, and λE is the error attributed to an error in $\frac{\partial T_t}{\partial t}$. Also, λ is a parameter $0 \leq \lambda \leq 1$ and ideally decaying with time.

The portion attributed to $T_1$ is directly used to compute a grid of errors in $T_1$, which is passed backward in time to the previous time step. The portion of error attributed to $\frac{\partial T_t}{\partial t}$ is directly used as the error in the output layer of the network and is backpropagated normally using the backpropagation algorithm and updates the weights in the network.

# 4 Results

## 4.1 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 was forecast on 37 levels of the atmosphere, this required training 74 different networks – two for each level for the U and V components.

## 4.2 Data Sets

The network was trained using the hourly input data sets to the Rapid Refresh (RAP) model for days divisible by 3 in January 2014 and validated against days 3n+1 in that same month.

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. This data can be downloaded from either the NCEP or NCDC ftp server [8,9].

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 subgrid 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 influenced by interaction with land, the planetary boundary layer. 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.

## 4.3 Analysis

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

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

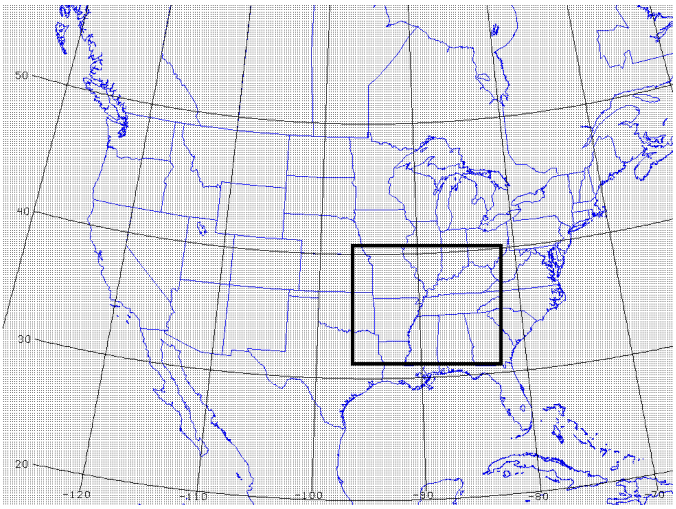


**Fig. 3.** NCEP Grid 252, a 301x225x37 grid of observations with approximately 20km horizontal resolution. Only the 56x49 grid over the southeastern US is used in this paper. This represents a roughly homogenous region with similar elevation and no ocean.

The forecasts generated by the proposed approach were compared to forecasts generated by the RAP model. The MAE of the 1 hour forecast generated by the RAP model is calculated for the same 56x49 sub-grid.

**Table 1.** Summary of results forecasting U and V for 10 levels of the atmosphere, where AIM3 represents the results of the proposed method. Error is MAE in m/s (meters per second).

| Level | RAP U | RAP V | AIM3 U | AIM3 V |
|---|---|---|---|---|
| 1000 | 0.4652 | 0.5219 | 0.4514 | 0.5807 |
| 900 | 0.7464 | 0.8519 | 1.0420 | 1.0868 |
| 800 | 0.7506 | 0.7588 | 1.0017 | 1.0764 |
| 700 | 0.7945 | 0.7715 | 1.0209 | 1.0714 |
| 600 | 0.8570 | 0.8695 | 1.1019 | 1.2007 |
| 500 | 1.0294 | 1.1761 | 1.2813 | 1.4829 |
| 400 | 1.3186 | 1.4488 | 1.8873 | 1.9040 |
| 300 | 1.5315 | 1.4577 | 2.0526 | 2.5082 |
| 200 | 1.2195 | 1.2825 | 1.5223 | 1.6046 |
| 100 | 0.7350 | 0.7379 | 0.9235 | 1.0091 |

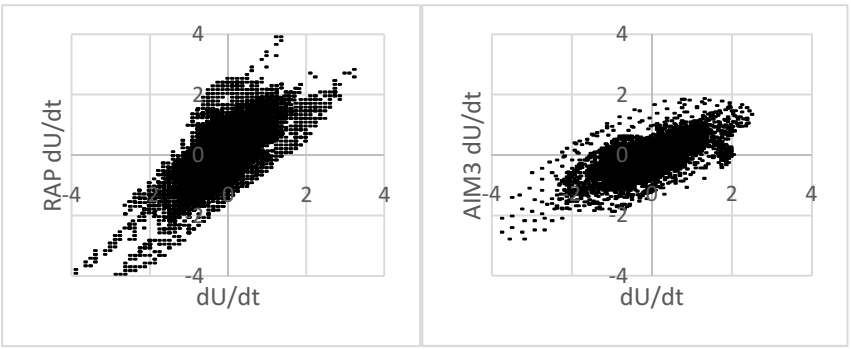


**Fig. 4.** Scatterplot for U Forecast on 1000mb Level

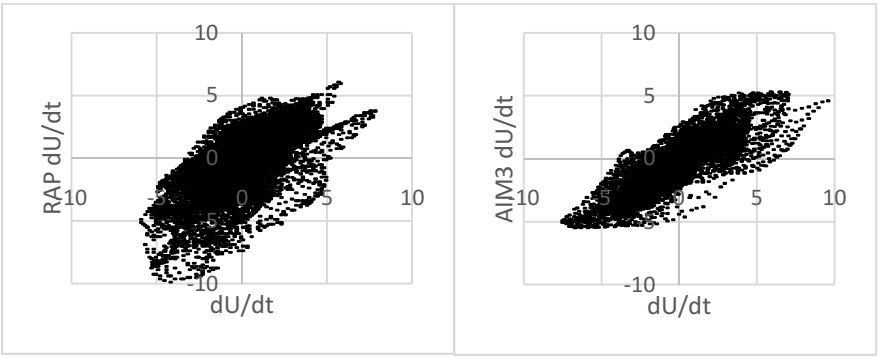


**Fig. 5.** Scatterplot for U Forecast on 500mb Level

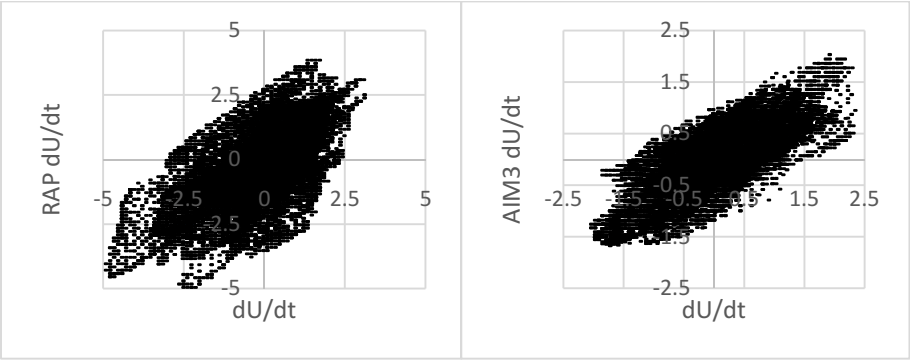


**Fig. 6.** Scatterplot for U Forecast on 100mb Level

## 5 Discussion

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

The implementation of the proposed method only forecasts horizontal wind speed. Vertical wind speed and geopotential height are calculated for each level using diagnostic equations. A complete model would forecast all model variables. Temperature and moisture forecasting are important components that 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 4, 5, and 6 show that the learned behavior closely mirrors the desired behavior.

Figure 4 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 at the surface, as can be seen in Table 1.

Figure 5 is the same as Figure 4, 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 6 is the same as Figure 5, 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 3$^{rd}$ order Runge-Kutta integration scheme, which is used by the RAP model [2]. The time step, 6 minutes in our implementation, could also be brought down to 1 minute to match the RAP model.

## 6 Conclusion

This method can be used to create a full AI-based meteorological 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.

Networks also need to be designed to forecast moisture transport, phase change, and latent heat. The remaining work there mainly involves selecting the appropriate inputs to consider. Although unimplemented, the network for forecasting temperature is given in Figure 2. With this, we would have a full forecast system.

Our implementation of the proposed approach has been successfully run out to 6 hours, but needs these additional components to generate competitive forecasts.

**Acknowledgements.** This work is partially supported by the Louisiana Board of Regents grant LEQSF-EPS(2013)-PFUND-307. We are grateful to Dr. Kevin Robbins from the NOAA Southern Regional Climate Center at LSU and Dr. Frederick Carr from the University of Oklahoma for many helpful discussions on topics related to this work.

## References

1. Stensrud, D.J.: Parameterization Schemes: Keys to Understanding Numerical Weather Prediction Models, pp. 7–9. Cambridge, New York (2007)
2. Coiffier, J.: Fundamentals of Numerical Weather Prediction, vol. 4-6, pp. 15–16. Cambridge, New York (2011)
3. Zakerinia, M., Ghaderi, S.F.: Short Term Wind Power Forecasting Using Time Series Neural Networks. University of Tehran, Tehran (2011)
4. Abdel-Aal, R.E.: Hourly temperature forecasting using abductive networks. Eng. App. of Art. Intel. (2004)
5. Mitchell, T.M.: Machine Learning, pp. 119–121. McGraw-Hill, Singapore (1997)
6. Krasnopolsky, V.M., Michael, S.F., Dmitry, V.C.: New Approach to Calculation of Atmospheric Model Physics: Accurate and Fast Neural Network Emulation of Longwave Radiation in a Climate Model. Mon. Wea. Rev. 133, 1370–1383 (2005)
7. Corne, D., Reynolds, A., Galloway, S., Owens, E., Peacock, A.: Short term wind speed forecasting with evolved neural networks. In: Blum, C. (ed.) 15th Genetic and Evolutionary Computation Conference Companion (GECCO 2013 Companion), pp. 1521–1528. ACM, New York (2013)
8. National Centers for Environmental Prediction, `ftp://ftp.ncep.noaa.gov/pub/data/nccf/com/rap/prod/`
9. National Climatic Data Center, `http://nomads.ncdc.noaa.gov/thredds/dodsC/rap252/`
10. Warner, T.T.: Numerical Weather and Climate Prediction, pp. 456–459. Cambridge, New York (2011)