Improving Forecasts of Geomagnetic Storms with Evolved Recurrent Neural Networks

Derrick T. Mirikitani, Lahcen Ouarbya, Lisa Tsui, and Eamonn Martin

***Abstract*— Recurrent neural networks (*RNN*s) have been used for modeling the dynamics of the D_{st} index. Researchers have experimented with various inputs to the model, and have found improvements in prediction accuracy using measurements of the interplanetary magnetic field (IMF) taken from the Advanced Composition Explorer satellite. The output of the model is the one hour ahead forecasted D_{st} index. Previous models have used gradient information, usually gradient descent, for optimization of *RNN* parameters. This paper uses the IMF inputs (that have been found to work well) to the *RNN* and uses a Genetic algorithm for training the *RNN*. The proposed model is compared to a model used in operational forecasts which relies on solar wind data and IMF parameters, as well as a model which uses IMF data only. Both of the comparison models were trained with gradient descent. A series of geomagnetic storms that so far have been difficult to forecast are used to evaluate model performance. It is shown that the proposed evolutionary method of training the *RNN* outperforms both models which were trained by gradient descent.**

I. Introduction

RECURRENT Neural Networks are some of the leading models used for prediction of geomagnetic storms [18], [19], [21]. The majority of the research in recurrent neural network based geomagnetic storm forecasting has focused on experimenting with various input quantities to the *RNN* to minimize forecast errors of the D_{st} index. This paper focuses on the training algorithms used for estimating the *RNN*s weights. Previous work in the area has relied on gradient based optimization [19], [21] which is susceptible to becoming trapped in local minima on the error surface, leading to suboptimal solutions. We propose the use of a genetic algorithm for estimation of recurrent neural network (*RNN*) parameters [4] rather than the gradient based method used exclusively in previous papers. Genetic algorithms use stochastic search operators, such as mutation and crossover, which allows the algorithm to move, from inside a local minima across a ridge to an even lower local minima with no more difficulty than descending directly into the local minima itself [16]. Genetic algorithms are efficient optimization procedures as they are able to reduce the forecast errors of the model monotonically. This study found that the use of the genetic algorithm can lead to improved out of sample performance on D_{st} forecasting tasks over models estimated with gradient descent. In the following sub-section we provide an overview of geomagnetospheric distrubances and a brief review of neural forecasting of geomagnetic storms.

The paper is organized as follows: Section II reviews geomagnetic storms, and neural network forecast models of the D_{st} index. Section III provides an overview of the Elman *RNN*, including a description of the architecture used in this paper. Section IV outlines the Genetic Algorithm used to estimate the weights of the RNN. Section V provides the results of a numerical simulation of the proposed model, and compares the results to those of other established models. Finally, Section VI concludes the paper.

II. Geomagnetic Storms

The magnetosphere is a magnetic field surrounding the Earth that shields and deflects charged particles emitted from the Sun from hitting the Earth. The sun also has a magnetic field, however the Sun's magnetic field is much more complex than the Earth's. It is well known that the variation in the Sun's magnetic field influences the structure of the magnetic field surrounding the Earth [2], [6], [12]. An Interplanetary Magnetic Field (IMF) is formed when the solar wind expands the reach of the Sun's magnetic field, which can extend to hit Earth's magnetic field. The IMF can cause energetic particles to enter into the Earth's magnetic field which results in magnetospheric disturbances. Disruption of the magnetosphere takes place when a transfer of energy from the solar wind opposes the Earth's magnetic field. A magnetospheric storm occurs if this transfer of energy persists for several hours [12].

Geomagnetic storms can have many negative effects on Earth resulting in widespread problems and damage to electric power grids, gas pipelines, power generation facilities, and Global Positioning System (GPS) disruption. Forecasting the earth's magnetic field can provide vital information about the intensity of future magnetospheric disturbances. At mid-latitudes, these magnetic storms are measured in relation to the horizontal component of the Earth's magnetic field [12]. The mean of this horizontal component is used to form an index known as the D_{st} index. There have been various studies that have shown a correlation between the value of the D_{st} index and the magnetic storm's intensity [8], [13], where the more negative the D_{st} index the greater the intensity of the magnetic storm. The physical interaction between the IMF and the magnetosphere takes place at the magnetopause boundary where a detailed understanding of this interaction is not yet fully understood. Previous researchers have built non-parametric predictive models based on recurrent neural networks (*RNN*s) to model this interaction [18], [21].

All authors are with the Department of Computing, Goldsmiths College, University of London, New Cross, London, UK (email: {D.T.Mirikitani, l.ouarbya}@gold.ac.uk).

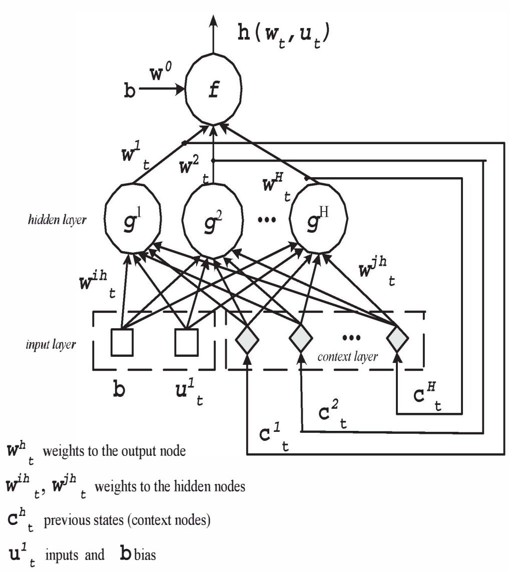

Fig. 1. the Elman recurrent neural network

Previous work in modeling the relationship between $\boldsymbol{IMF}$ and $\boldsymbol{D_{st}}$ with *RNN*s have relied heavily on first-order gradient based methods for parameter estimation of the model [17], [21] which has resulted in long training times, uncertain convergence, and possibly vanishing gradients. In this paper we investigate solutions to this problem through the use of the genetic algorithms for *RNN* training [4]. The advantage of our approach is a framework based on global search strategies resulting in superior convergence and accurate forecasts. The main results of the paper are as follows: 1) a stochastic global search strategy for *RNN* parameter estimation for $\boldsymbol{D_{st}}$ forecasting, 2) improved forecast accuracy over previously demonstrated results.

III. The Elman Recurrent Neural Network

In this study, the recurrent architecture known as the Elman network (*RNN*) [7] is chosen as previous studies have found successful results with *RNN*s. Feed-forward networks are not considered in this study due to poor performance in modeling the recovery phase dynamics [9]. This is most likely due to the limitation of the feed-forward architecture, i.e. limited temporal memory, bounded by the dimension of the input window.

The *Elman network* is essentially a feed forward multi layer perceptron with a context layer. The context layer copies the hidden layer activations and feeds it back in to the hidden layer one time step later. This simple addition allows for temporal processing due to the circular flow of information.

Elman networks have been used in many applications for processing time series data [1]. Various studies have found that the Elman network is able to outperform the feed-forward network on time series forecasting tasks [10].

A system of difference equations can be used to compute the hidden layer activation:

$$s^{(k)} = g^{(k)}(\mathbf{u}_t, \mathbf{s}_{t-1}) \tag{1}$$

The above equation describes the operation of the k-th hidden network node at time t. The activation function $g(\cdot,\cdot)$ is a nonlinear sigmoidal function of the input vector $\mathbf{u}_t$ and the context vector $\mathbf{s}_{t-1}$. The network output is computed via

$$y_t = \sigma(f(\mathbf{s}_t)) \tag{2}$$

where $\mathbf{s}_t$ is the vector of hidden activations

$$\mathbf{s}_t = [s^{(1)}, s^{(2)}, \ldots, s^{(k)}, \ldots, s^{(H)}] \tag{3}$$

and the context vector

$$\mathbf{c}_t = [c^{(1)}, c^{(2)}, \ldots, c^{(k)}, \ldots, c^{(H)}] = \mathbf{s}_{t-1} \tag{4}$$

is a vector of previous states, where H is the number of hidden neurons.

Figure 1 provides a diagram of the Elman network with bias term $b^{(1)}$ and a single network input $\mathbf{u}_t$. To simplify notation, the function $z_t(i)$ is introduced to represent the bias, input, or context activation.

$$z_t(i) = \begin{cases} b, & i = 0 \\ \mathbf{u}_t^{(j)}, & 1 \leq i \leq L \\ \mathbf{c}_t^{(i-L)}, & L+1 \leq i \leq L+H \end{cases} \tag{5}$$

In the hidden layer, each neuron computes its activation $s_t^{(k)}$ via the following equation:

$$\begin{aligned} \acute{s}_t^{(k)} &= \sum_{j=0}^{L+H} z_t(j) w^{(k,j)} \\ &= b^{(1)} w^{(k,0)} + \sum_{j=1}^{L} w^{(k,j)} \mathbf{u}_t^{(j)} + \sum_{i=L+1}^{L+H} w^{(k,i)} \mathbf{c}_t^{(i-L)} \end{aligned} \tag{6}$$

$$s_t^{(k)} = g^{(k)}(\mathbf{u}_t, \mathbf{s}_{t-1}) = \sigma(\acute{s}_t^{(k)}) \tag{7}$$

where the bias weight is $w^{(k,0)}$, the input weights are $w^{(k,j)}$ for $1 \leq j \leq L$, and the context weights are $w^{(k,i)}$ for $1 \leq i \leq L+H$ which connect the corresponding past i-th hidden node output to the current state input. The activation functions $\sigma(\cdot)$ are sigmoidal nonlinearities

$$\sigma(a) = 1/(1 + \exp(-a)) \tag{8}$$

which map the input a from $\mathcal{R}$ into a bounded interval $\Omega = (0,1)$ of length $|\Omega| = 1$ where $\Omega \subset \mathcal{R}$.

The network output is a combination of the sum of the states multiplied by the corresponding weights along with the bias. The sum is passed through the activation function to yield the network output:

$$y_t = \sigma\left(b^{(0)} w^{(0)} + \sum_{j=1}^{H} w_o^{(j)} s_t^{(j)} \right) = \sigma(f(\mathbf{s}_t)) \tag{9}$$

The output bias and corresponding weight is given by $b^{(0)}$ and $w^{(0)}$ respectively. The hidden to output node weights $w_o^{(j)}$ connect the states of the network to the output node y_t. The total number of weights in the model is $m = H + 1 + (H + I + 1) \times H$. The overall network model thus becomes a highly nonlinear function $h(\mathbf{w}_t, \mathbf{u}_t)$ of the weights $\mathbf{w}_t$ and inputs $\mathbf{u}_t$.

$$\begin{aligned} d_t &= h(\mathbf{w}_t, \mathbf{u}_t) + \epsilon_t \\ &= y_t + \epsilon_t \end{aligned} \tag{10}$$

where the noise ϵ_t is assumed to be independent zero-mean Gaussian with variance σ_y^2 which is unknown: $\epsilon_t \sim \mathcal{N}(0, \sigma_y^2)$.

IV. Evolutionary Training of the *RNN*

The $\boldsymbol{D_{st}}$ forecasting literature has relied on gradient descent optimization for estimation of model parameters (weights) of the *RNN*. To compute the gradient of the cost function with respect to each model parameter in neural networks, backpropagation is commonly used [22], [23]. However, there are drawbacks to the gradient based approach to training *RNN*s; vanishing gradients [3], [15], high computational complexity [24], local minima [3], [5]. These problems make gradient based training difficult and most likely leads to poor out of sample forecasts. Global search strategies such as genetic algorithms [14] are known to be effective in addressing these limitations.

Genetic algorithms (GAs) are stochastic global optimization procedures inspired by the process of evolution found in nature [11], [16]. Unlike gradient descent learning which iteratively improves a single solution (i.e. one *RNN* weight vector), the GA maintains population of individuals (i.e. a set of *RNN* weight vectors) that effectively samples the error surface at multiple points. The maintenance of a population allows the GA to be less susceptible to getting trapped in a local minima than gradient based methods.

As evolution works on populations of individuals, the algorithm starts off by randomly generating a population of chromosomes (*RNN*s). Each chromosome in the population represents an *RNN* which can compute a solution to the forecasting problem. Genetic algorithms can promote learning or adaptation in *RNN*s through evolution of the *RNN* model parameters (i.e. *RNN* weights). This is typically accomplished by encoding the *RNN* model parameters into chromosome like structures where each weight represents a gene on the chromosome. The evolutionary process favors solutions that carry higher relative fitness. In the genetic algorithm, it is the chromosomes with higher fitness that are more likely to be selected for reproduction, leading to the survival of organisms with fitter characteristics in the next generation. Here we favor individuals which have lower error (higher fitness) on the training set (in-sample data). Inspired by our understanding of evolution in nature, each genotype (weight vector) is transformed into its phenotypical representation (the *RNN*) and evaluated on the in-sample data to assess its fitness. The individuals (genotypes) which perform better, i.e. have lower error on the in-sample data are given a higher fitness than those which do not perform as well. The higher fitness translates into a higher probability to be selected for reproduction. Roulette wheel selection is used to probabilistically choose the parents that will create the offspring that survive in the next generation. The offspring are created by applying the standard genetic operators, crossover and mutation, on the probabilistically selected parents. The evolutionary operators of crossover and mutation are invoked on two selected parents to produce new offspring. The crossover operator takes two parents and produces two offspring by cutting and concatenating chromosome segments from both parents. After selecting a random position on the chromosome, both parents chromosomes are cut and the front portion of the first parent's chromosome is appended to the rear portion of the second parents chromosome, and the front portion of the second parent's chromosome is spliced to the rear portion of the first parent's chromosome. The mutation operator can then be applied to the offspring. During mutation, one of the alleles on the chromosome is randomly changed to a new value. The offspring are then placed into a population representing the next generation. Performing these steps over multiple generations tends to lead to successive populations of increasing fitness, converging toward the globally optimal solution.

The genetic algorithm for training *RNN*s can be summarized as follows:

1) Randomly generate the initial population of chromosomes.
2) Compute the fitness of each member of the population by running each *RNN* over the training data and measuring the error.
3) If at least one of the members of the population have higher fitness (lower MSE) than the predetermined requirements then stop. Otherwise, continue on to the next step.
4) Copy the chromosome with the highest fitness into the population of the next generation (elitism).
5) Continue to fill the population of the next generation by applying the selection operator to select the parents and then apply the genetic operators of crossover and mutation.
6) Go to step 2

Through these steps, the genetic algorithm finds a set of weights that minimizes the error between the training targets and the network output.

A. Fitness Function

The fitness function is a fundamental component of the GA, which is used to distinguish the useful solutions from less useful solutions in a population. Each chromosome in the population is evaluated by transforming each chromosome into an *RNN*, i.e. each weight in the chromosome is placed into its corresponding position in the *RNN* connectivity graph. Then, the training set is fed into the *RNN* and the one step ahead forecasts from the *RNN* are measured for accuracy. The accuracy of each *RNN* is measured via the MSE error function:

$$\epsilon^{(i)} = \frac{1}{T}\sum_{k=1}^{T}(d_k^{(i)} - y_k)^2 \tag{11}$$

which takes the difference between the output of the ith *RNN* $d_k^{(i)}$ at data point k and the corresponding target y_k. The differences are squared, and the sum of these differences is computed and averaged over the length T of the training set.

B. Cross Over

Crossover is the mechanism by which genetic information from two selected parents is recombined to from new offspring with potentially better fitness. It is thought that by recombining genetic information from fit individuals, even fitter offspring can be generated. Through crossover, the search becomes focused toward regions of higher fitness. Crossover is implemented by selecting two parent chromosomes to form two children. A point on the chromosome $i \sim \mathcal{U}(1, \mathfrak{n}-1)$, is randomly drawn from the uniform distribution which demarcates where the cutting and swapping of genetic material will take place.

Given two selected chromosomes $\mathbf{C}_1 = (c_1^1, \ldots, c_1^{\mathfrak{n}})$ and $\mathbf{C}_2 = (c_2^1, \ldots, c_2^{\mathfrak{n}})$, the genetic splicing takes place via:

$$\begin{aligned} \acute{\mathbf{C}}_1 &= \{c_1^1, c_1^2, \ldots, c_1^i, c_2^{i+1}, \ldots, c_2^{\mathfrak{n}}\} \\ \acute{\mathbf{C}}_2 &= \{c_2^1, c_2^2 \ldots, c_2^i, c_1^{i+1}, \ldots, c_1^{\mathfrak{n}}\} \end{aligned} \tag{12}$$

resulting in two new individuals for the next generation.

C. Mutation

The mutation operator promotes diversity in the population, allowing for exploration of areas of the search space not bounded by the population. Given a chromosome chosen for mutation, $\acute{\mathbf{C}}_1 = \{c_1^1, \ldots, c_1^{\mathfrak{n}}\}$, a point c^i on the chromosome is chosen from the uniform distribution $i \sim \mathcal{U}(1, \mathfrak{n})$. Mutation of the gene c^i happens by adding by a Gaussian random number $r \sim \mathcal{N}(0, \sigma)$ to itself:

$$c_{new}^i = c^i + r \tag{13}$$

and then replacing the new gene c_{new}^i with the old gene c^i.

V. Experimental Results

The performance on out of sample data of the proposed algorithm and both Lundstedt's algorithm [19], [20] and the Edda algorithm are provided in this section. The model proposed by Lundstedt et. al., uses an $\boldsymbol{IMF}$ component ($\boldsymbol{b_z}$) and solar wind parameters ($\boldsymbol{n}, \boldsymbol{v}$) as inputs to the *RNN* to forecast the $\boldsymbol{D_{st}}$ index. The Edda algorithm proposed by Pallocchia et. al. [21], uses the same recurrent neural architecture as in Lundstedt (i.e. an *RNN*), and uses the same gradient based optimization scheme as in Lundstedt. The difference between the Edda algorithm and Lundstedt's algorithm is in the inputs to the RNN. The Edda algorithm omits the solar wind measurements because they were found to be unreliable during geomagnetic storms (solar wind data is collected through satellite based sensors that tend to malfunction during geomagnetic storms). The Edda algorithm uses instead the IMF parameters $\boldsymbol{b}$, $\boldsymbol{b_y}$ and $\boldsymbol{b_z}$ which are measured from the ground [21].

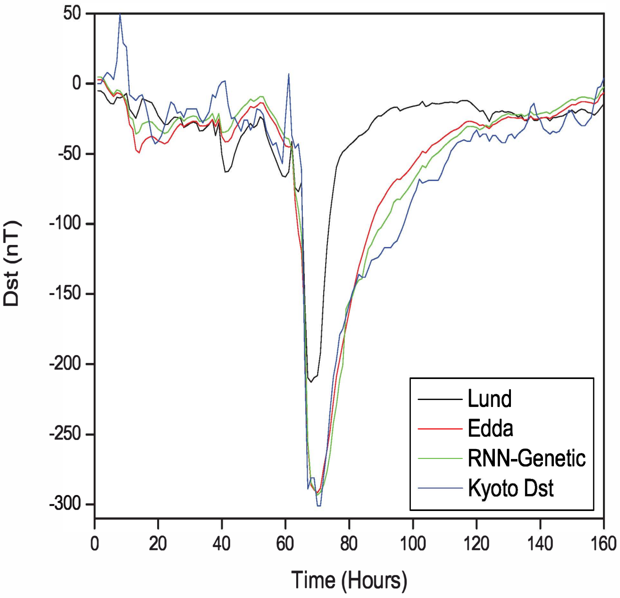

Fig. 2. prediction of $\boldsymbol{D_{st}}$ from July 13th - July 19th 2000.

The inputs for Lundstedt's algorithm were transformed in the following way: $u^{(1)}(t) = \boldsymbol{b_z}(t)/30$, $u^{(2)}(t) = \boldsymbol{n}(t)/80 - 1$, $u^{(3)}(t) = \boldsymbol{v}(t)/400 - 1.5$. The output of Lundstedt's *RNN* was scaled by $\hat{\boldsymbol{D_{st}}}(t+1) = 150y(t) - 100$ [20]. The inputs to the Edda algorithm were transformed by the following formula: $u^{(1)}(t) = \boldsymbol{b_z}(t)/44.7$, $u^{(2)}(t) = (\boldsymbol{b}(t))^2/(48.2)^2$, $u^{(3)}(t) = (\boldsymbol{b_y}(t))^2/(34.4)^2$, and the output of the *RNN* was scaled by $\hat{\boldsymbol{D_{st}}}(t+1) = y(t)387$ [21]. The proposed genetic algorithm trained *RNN* uses the same transformed inputs and scaled output as the Edda algorithm.

The training set was constructed from 8 storms from the dates of 01-01-1998 to 07-12-2001. To speed up training, the quiet periods were largely omitted. The evolutionary algorithm was allowed to run until there was an average of 10nT error per hour.

In the following subsections, the performance of the proposed model is compared to the Lund and Edda models on a variety of storms including a severe storm and two double storms.

A. Storm 1, Severe Storm

Figure 2 illustrates the performance of the models on out of sample data of a storm which occurred between the dates of July 13th and July 19th 2000. The blue line represents the Kyoto $\boldsymbol{D_{st}}$ index values. The black line represents the one hour ahead forecasts from the Lund model, the red line represents the one hour ahead forecasts from the Edda model, and the green line represents the one hour ahead forecasts from the proposed model.

The storm starts off with three large compressions before dropping severely to -300 nT. The D_{st} index recovers quickly at first but then at around -150 nT, the storm's recovery begins to slow and takes nearly 80 hours for the D_{st} index to reach normal levels.

All three models are unable to capture the initial three compressions. However, once the D_{st} index starts dropping off, all three models are able to recognize that the storm is beginning. The forecasts are nearly identical for the first few hours of the initial phase (the drop in D_{st}) of the storm. The Lund algorithm is the first model to deviate from the D_{st} index. It begins to diverge when the D_{st} index passes -200 nT. After a few hours the Lund algorithm begins to predict the storm is going into the recovery phase and starts to forecast increasing D_{st} values. The Edda and the proposed model (*RNN*-Genetic) continue to forecast accurate values of the downward phase of the storm. The first few hours of the recovery phase of the storm are forecasted accurately by both the Edda and the proposed model. The divergence between the two models begins at approximately 78 hours into the storm, at a point when the recovery of the storm begins to slow down. Both models begin to overstate the recovery phase, with the Edda model producing less accurate values than the proposed model. The forecast errors of each model on the first storm are given in the second row of Table 1. The proposed *RNN*-Genetic model significantly outperforms the Lund model and has slightly lower forecast errors than the Edda model.

B. Storm 2, Double Storm

Figure 3 presents the forecasts of the three models on a double storm that occurred between August 9th and August 16th 2000. The storm starts off with an initial decrease in the D_{st} index, but a recovery begins when the D_{st} index hits around -100 nT. The recovery is short lived, and after about 16 hours, the D_{st} index begins to decrease sharply until reaching nearly -238 nT. Once hitting its minimum, the storm begins an immediate recovery, with the D_{st} index rapidly climbing to a value of nearly -60 nT before slowing. From this point, it takes the storm an additional 80 hours before a full recovery is made.

The Lund model severely underestimates the first and second drop in the D_{st} index. The Lund model also overestimates the recovery phase of the storm, between hours 60 and 120 in Figure 3. The Edda algorithm (red line) accurately predicts the initial drops of the two storms. However, it underestimates the recovery phase of both storms. The proposed model, the RNN-Genetic (green line) has similar behavior to the Edda model in modeling the drop of both storms. The difference between the two models becomes more apparent in the recovery phase of the storms. The RNN-Genetic model provides a slightly more accurate estimation of the recovery phase of both storms as shown by the green line in Figure 3. The third row of Table 1 shows the performance of the three models in terms of root mean squared error (RMSE). The proposed algorithm (RNN-Genetic) outperforms the gradient based algorithm of Lund in all phases of the storm resulting in significantly less errors. The proposed model also outperforms the Edda model mainly during the recovery phase of the storms resulting in a reduction in forecast errors over the Edda model. It may be worth pointing out that the Lund model outperforms the Edda model in this example, but the proposed model, although similar in behavior to the Edda algorithm outperforms both models on this storm.

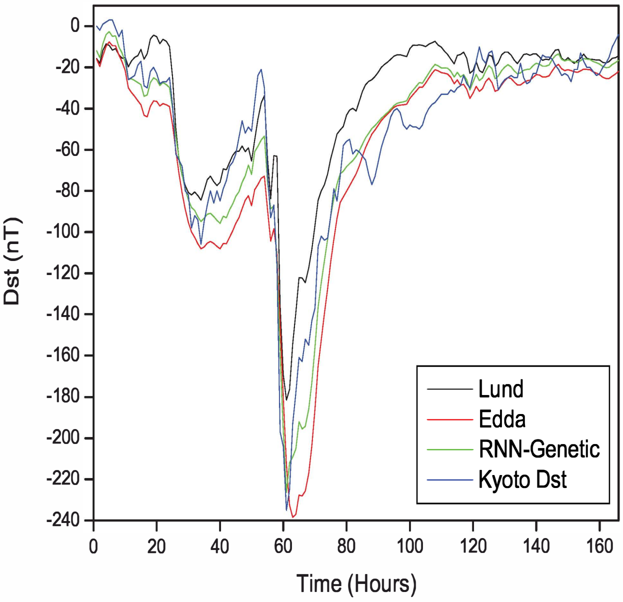

Fig. 3. prediction of D_{st} from August 9th - August 16th 2000.

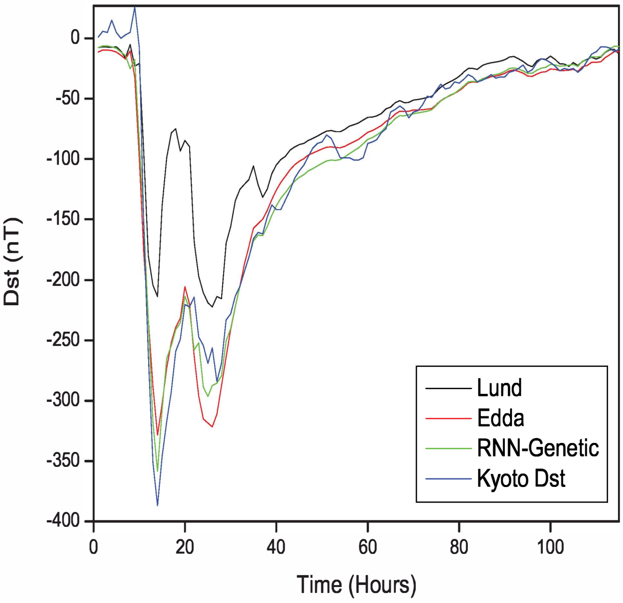

Fig. 4. prediction of D_{st} from March 30th - April 5th 2001.

C. Storm 3, Severe Double Storm

Figure 4 presents the final storm of the experimental section which occurred between the dates of March 30th - April 5th 2001. Before the downward phase of the storm,

TABLE I

PERFORMANCE OF RNN MODELS ON OUT OF SAMPLE DATA

Type	Lund RMSE	Edda RMSE	Genetic RMSE
Storm 1	52.90	20.78	17.45
Storm 2	21.72	22.31	13.37
Storm 3	56.21	21.28	15.91

there are two initial compressions that take place. The second compression is followed immediately by a decrease in the D_{st} index to a value of around -375 nT. The D_{st} index then rebounds sharply to nearly -210 nT before shifting directions back downwards to a second minimum of about -280 nT. This minimum is followed by a characteristic recovery phase, rapid in the beginning, but gradually slowing down toward the end of the storm.

All models fail to capture the first compression, but the Lund and Edda models have a very slight upward spike representing the second compression. The proposed *RNN*-Genetic model fails to represent these compressions. All three models accurately represent the initial moments of the downward movement of the storm. However, the Lund model begins the rebound in D_{st} far too early at a value near -200 nT. As the descent in D_{st} progresses, the Edda model also begins its rebound nearly 75 nT too early while the proposed model continues its descent before rebounding just past -350 nT. The rebounds of both the Edda and the proposed model are similar but the Edda model overshoots on the second dip in the D_{st}. Figure 3 highlights how the genetic algorithm reproduces a better representation of the target line of the storm compared to both the Lund and Edda algorithms. In the final row of Table 1, the performance of all models in terms of RMSE are presented for the severe double storm.

VI. FUTURE WORK AND CONCLUDING REMARKS

This paper provides a method to improve forecast models of the D_{st} index through the use of IMF data and genetic algorithms for training *RNN* models of geomagnetic storms. It was found that the genetic algorithm can reduce forecast errors of the D_{st} index in comparison to similar models trained with gradient based methods [1], [19]. It is thought that global stochastic search has led to the reduction in D_{st} forecast errors of the *RNN* as it is less susceptible to local minima, and avoids vanishing gradients common in the computation of *RNN* derivatives [3]. This paper offers an alternative to gradient based learning for *RNN* based D_{st} forecasting.

Although *RNN*s have become popular forecast models of the D_{st} index [1], [19], [21], nearly all research has utilized gradient based training algorithms for estimation of *RNN* parameters. This paper looked into an alternative method for estimating model parameters. The results have shown that forecast errors can be reduced through evolutionary *RNN* training. However, evolution only worked on the weights of the neural network, and not on the topology, the inputs, the delays between inputs, etc. Future work will look into whether optimized architectures can also lead to improvements in forecasting the D_{st} index.

REFERENCES

[1] E. Amata, G. Pallocchia, G. Consolini, M. F. Marcucci, and I. Bertello. Comparison between three algorithms for dst predictions over the 2003-2005 period. Journal of Atmospheric and Solar-Terrestrial Physics, 70, 496-502, 2008.

[2] W. I. Axford and C .O. Hines. A unifying theory of high-latitude geophysical phenomena and geomagnetic storms. Can. J. Phys. 39, 1433–1464, 1961.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157166, 1994.

[4] A. Blanco, M. Delgado, M.C. Pegalajar. A real-coded genetic algorithm for training recurrent neural networks. Neural Networks. 14, 93–105, 2001.

[5] C. M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 1995

[6] J. W. Dungey. Interplanetary magnetic field and the auroral zones. Phys. Rev. Lett. 26, 47–48, 2000.

[7] J. L. Elman. Finding Structure in Time. Cognitive Science, 14, 179–211, 1990.

[8] C. J. Farrugia, M. P. Freeman, L. F. Burlaga, R. P. Lepping, and K. Takahashi. The earth's magnetosphere under continued forcing - Substorm activity during the passage of an interplanetary magnetic cloud. J. Geophys. Res. 98, 7657–7671, 1993.

[9] H. Gleisner, H. Lundstedt, and P. Wintoft. Predicting Geomagnetic Storms From Solar-Wind Data Using Time-Delay Neural Networks. Ann. Geophys. 14, 679–686, 1996.

[10] R. Gencay, and T. Liu. Nonlinear modeling and prediction with feedforward and recurrent networks. Physica D, 108(1),119-134, 1997.

[11] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning Addison-Wesley Pub. Co. 1989.

[12] W. D. Gonzales, J. A. Joselyn, Y. Kamide, H. W. Kroehl, G. Rostoker, B. T. Tsurutani, and V. M. Vasyliunas. What is a geomagnetic storm? J. Geophys. Res. 99, 5771–5792, 1994.

[13] J. T. Gosling, D. J. McComas, J. L. Phillips, S. J. Bame. Geomagnetic activity associated with earth passage of interplanetary shock disturbances and coronal mass ejections. J. Geophys. Res. 96, 7831–7839, 1991.

[14] J N. D. Gupta and R. S. Sexton. Comparing backpropagation with a genetic algorithm for neural network training. Omega, 27(6),679–684, 1999

[15] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, 2001.

[16] J. H. Holland. Adaptation in natural and artificial systems MIT Press Cambridge, MA, USA, 1992.

[17] H. Lundstedt. Neural Networks and prediction of solar-terrestrial effects. Planet. Space Sci. 40, 457–464, 1992.

[18] H. Lundstedt and P. Wintoft. Prediction of geomagnetic storms from solar wind data with the use of a neural network. Ann Geophys. 12, 19–24, 1994.

[19] H. Lundstedt, H. Gleisner, and P. Wintoft. Operational forecasts of the geomagnetic Dst index. Geophys. Res. Lett. 29, 34–1–34–4, 2002.

[20] Lund Space Weather Center, `http://www.lund.irf.se/rwc/dst/models/`

[21] G. Pallocchia, E. Amata, G. Consolini, M. F. Marcucci and I. Bertello. Geomagnetic Dst index forecast based on IMF data only. Ann Geophys. 24, 989–999, 2006.

[22] P. J. Werbos. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis, Harvard University, 1974.

[23] P. J. Werbos, Backpropagation Through Time: What it does and how to do it, Proceedings of the IEEE 78, 1550–1560, 1990

[24] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Backpropagation: Theory, Architectures and Applications. Hillsdale, NJ: Erlbaum, 1994