Toward Automatic Time-Series Forecasting Using Neural Networks

Weizhong Yan, *Senior Member, IEEE*

***Abstract*— Over the past few decades, application of artificial neural networks (ANN) to time-series forecasting (TSF) has been growing rapidly due to several unique features of ANN models. However, to date, a consistent ANN performance over different studies has not been achieved. Many factors contribute to the inconsistency in the performance of neural network models. One such factor is that ANN modeling involves determining a large number of design parameters, and the current design practice is essentially heuristic and ad hoc, this does not exploit the full potential of neural networks. Systematic ANN modeling processes and strategies for TSF are, therefore, greatly needed. Motivated by this need, this paper attempts to develop an automatic ANN modeling scheme. It is based on the generalized regression neural network (GRNN), a special type of neural network. By taking advantage of several GRNN properties (i.e., a single design parameter and fast learning) and by incorporating several design strategies (e.g., fusing multiple GRNNs), we have been able to make the proposed modeling scheme to be effective for modeling large-scale business time series. The initial model was entered into the NN3 time-series competition. It was awarded the best prediction on the reduced dataset among approximately 60 different models submitted by scholars worldwide.**

***Index Terms*— Artificial neural network, generalized regression neural network, model combination, time-series forecasting.**

I. Introduction

CONVENTIONALLY, time-series forecasting (TSF) has been performed predominantly using statistical-based methods, for example, the autoregressive integrated moving average (ARIMA). However, over the past few decades, artificial neural networks (ANNs)—which exhibit superior performance on classification and regression problems in machine-learning domain—have attracted tremendous attention in the TSF community. Compared to statistics-based forecasting techniques, neural network approaches have several unique characteristics, including: 1) being both nonlinear and data driven; 2) having no requirement for an explicit underlying model (nonparametric); and 3) being more flexible and universal, thus applicable to more complicated models [1].

Because of the aforementioned characteristics, ANNs have been regarded by many experts as a promising technology for TSF. Consequently, in the last few decades, more than 2000 articles on neural network forecasting have been published, covering a wide range of applications/fields [2]. A great number of empirical studies have shown superior performance of neural network forecasters over statistical methods, based on a single or a small set of time series (e.g., [3]–[5]). However, several studies have shown inferior performance of neural network methods over traditional statistical methods. Heravi *et al.* [6] conducted a study on comparing neural network models against linear autoregressive models based on 24 time series of annual change in monthly industrial production in three European countries. Their results indicated that linear models generally outperform ANN models. A study by Callen *et al.* [7], based on 296 quarterly accounting-earning series, also showed the inferiority of neural networks over linear models.

Several factors contribute to the inconsistent results of neural network models across different studies. Adya and Collopy [8] attributed the inconsistency to the ineffectiveness in validation and implementation involved in some studies. They found that, out of 48 business-related forecasting studies they identified, only 11 (i.e., 22.9%) were correctly implemented and validated. On the other hand, Nelson *et al.* [9] and Zhang and Kline [10] suggested that the inconsistency of ANN performance from different studies was the result of different preprocessing strategies adopted in those studies. They believe that time-series preprocessing (e.g., detrending and deseasonalizing) contributes significantly to ANN model performance.

Others (e.g., [11]) attributed the inferior performance of ANN models to the inherent requirement of a sufficient number of samples in order for networks to be fully trained. Most real-world applications, especially economic and financial series, however, are short ones, which are inadequate for ANNs to learn the underlying pattern and structure, thus resulting in poor out-of-sample prediction performance. Still, several studies, for example, [1] and [12], attributed the poor performance of ANNs to the need for determining a large number of network parameters (e.g., network types, architectures, and many other network parameters); the current heuristic and largely ad hoc modeling process does not allow for exploiting the full potential of ANN models.

Clearly, how effective ANNs are for TSF is still an open question and more research work is needed in this regard. One approach to maximally exploring the potential of ANN models for TSF is through organized TSF competitions. By pooling the development effort and skills from worldwide participants, a large number of empirical studies can be performed effectively. The M3 Competition [13] and more

Manuscript received January 15, 2012; revised April 19, 2012; accepted April 20, 2012. Date of publication June 1, 2012; date of current version June 8, 2012.

The author is with the Machine Learning Laboratory, GE Global Research Center, Niskayuna, NY 12309 USA (e-mail: yan@ge.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2198074

recently the Neural Network Forecasting Competition [12] are two examples. Another important research effort is on the development of a systematic modeling approach to replace the current heuristic and ad hoc methods. By doing that, we can alleviate the difficulty encountered in ANN modeling, and thus allow for a more reliable evaluation of ANN models for a large number of time series. To that end, Balkin and Ord [11] proposed an automatic procedure for neural network modeling. Their work, however, was limited to multilayer perceptron (MLP) networks only and their method showed disappointing performance in the M3 competition.

Our effort in this paper is also toward automatic ANN modeling for TSF. Realizing that MLP networks have several shortcomings (e.g., a large number of design parameters, long training time, and suffering from local minima) that make model automation more difficult, in this paper we attempt to develop an automatic modeling scheme using generalized regression neural networks (GRNNs), a special type of neural networks. GRNN has only a single design parameter and is simple and fast in training, which are the desirable properties for automated modeling. Our effort in this paper focuses on designing a modeling scheme to take full advantage of these GRNN properties. In addition, we pay great attention to the following design strategies to achieve a more effective and efficient automation and good forecasting performance as well:

1) using fusion of multiple GRNNs to alleviate the difficulty of determining the spread factor of GRNN;
2) using automatic feature identification and treatment strategy for outliers and trends;
3) adopting the direct approach for multiple-step-ahead (MSA) forecasting.

Our initial model was entered in the 2007 Neural Network Time-Series Forecasting Competition (www.neural-forecasting-competition.com). Among approximately 60 submissions to the competition, our model was the overall winner for the best prediction of the reduced dataset (11 time series)[1] based on several different error measures [12].

The rest of this paper is organized as follows. Section II provides an overview of related work on neural network forecasting. Section III gives a brief introduction to GRNN. The proposed ANN modeling scheme is described in detail in Section IV. Section V gives our experimental results of the model. Section VI is dedicated to the sensitivity study and discussion. Section VII concludes this paper.

II. Related Work

In the literature, there are a large number of publications on using neural networks for TSF. These publications cover a wide range of TSF applications, varying from financial [7] to economic [4], to natural physical phenomena—for example, river flow [14], earthquakes [15], and weather [16]. Several overview papers, for example, [1], [8], [17], provide a good summary of various applications of neural networks for time series forecasting.

Many types of networks have been employed for those diverse TSF applications. The earliest and the most popular type of networks is feed-forward MLPs. Early applications of this type of networks include the paper by Farway and Chatfield [18], Hill *et al.* [3], and Balkin and Ord [11], while studies by Zhang and Kline [10] and Heravi *et al.* [6] are some examples of more recent work. Recurrent neural networks are another type of networks that are often used for TSF (e.g., [19]–[22]). Other types of neural networks used for TSF include radial basis function (RBF) networks [23], Bayesian neural networks [24], and neuron-fuzzy networks [25], [26]. Ahmed *et al.* [27] conducted a comparison study of different types of machine-learning models, including MLP, Bayesian neural networks, RBFs, GRNNs, k-nearest neighbor regression, regression tree, support vector regression, and Gaussian processes. Recently, extreme learning machine (ELM), a new type of neural networks introduced by Huang *et al.* [28], was adopted by Sorjamaa *et al.* [29] for TSF. More recently, using Clifford support vector machines [30] and causality analysis [31] for TSF has also been investigated. GRNNs have also been used for TSF. For related work on GRNNs, please see Section III for details.

In addition to single models, multiple models have also been used for TSF, which is often referred to as *combining forecasts* [32]. Combining multiple statistical models for forecasting has a long history [33]. Empirical studies have shown that combining forecasts not only improves forecasting accuracy but also alleviates the difficulty associated with the conventional design strategy of selecting a single model. For example, Zou and Yang [34] convexly combined multiple ARIMA models with different orders using the so-called aggregated forecast through exponential reweighting (AFTER) scheme and showed that model combination could perform better than model selection when uncertainty in model selection is significant.

Combining multiple neural network models for forecasting has also been actively explored in recent years. Both theoretical and empirical studies have shown that by intelligently combining the outputs from a series of networks, greater accuracy than the best individual network can be achieved. Inspired by the success of neural network ensembles (NNEs) first introduced by Hansen and Salamon [35] in the domains of machine learning and data mining, several studies used the concept of the NNE for TSF. For example, Maqsood *et al.* [16] used an ensemble of different types of networks for weather forecasting in southern Saskatchewan, Canada. Lim and Goh [36] proposed a modified version of the AdaBoost algorithm to integrate the predictions of multiple Elman recurrent networks. Original AdaBoost works for classification tasks only, modified AdaBoost works for regression problems so that it can be used for TSF. More recently, Assaad *et al.* [37] also studied boosting multiple recurrent neural networks for single-step-ahead (SSA) and MSA prediction problems. Both boosting studies, however, were limited to benchmarking datasets (e.g., the sunspots and the Mackey–Glass time series) and have not yet been applied to real-world problems.

[1]For the complete dataset, our initial model was ranked 11th in the competition and our improved model that is described in this paper is ranked 5th overall (See Table I in Section V).

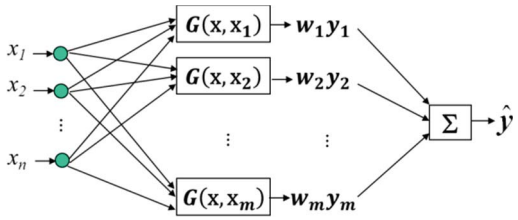

Fig. 1. Typical GRNN structure.

III. GRNNs

GRNN is a variant of RBF networks [38, p. 277]. A typical GRNN has three layers of artificial neurons (see Fig. 1): the input layer, the hidden layer, and the output layer. The hidden layer has radial basis neurons, while neurons in the output layer have a linear transfer function. Given a sufficient number of neurons, GRNN can approximate a continuous function to an arbitrary accuracy (i.e., it is a universal approximator) [39].

In the original GRNN, the number of radial basis neurons in the hidden layer is set to equal the number of training samples. In such design, a large number of training samples will result in a complex network with too many hidden neurons. One way of reducing the number of hidden neurons is to cluster the training samples first and to set the number of hidden neurons equal to the number of clusters [40]. The input to each of the radial basis neurons is the distance between the input vector and the training sample. The neuron's output is the RBF of the input scaled by the spread factor. This type of neuron gives an output characterizing the closeness between input vectors and training samples. The commonly used RBF is the multivariate Gaussian function defined by

$$G(x, x_i) = \exp\left(-\frac{1}{2\sigma^2}||x - x_i||^2\right) \tag{1}$$

where x_i and σ are the center and width, respectively, of the Gaussian function.

Given m input–output pairs $\{x_i, y_j\} \epsilon \Re^n \times \Re^1, i = 1, 2, \ldots, m$, and as the training samples, assume the original design of GRNN, that is, the number of hidden neurons is equal to the number of training samples. The GRNN output for a test point, $x \epsilon \Re^n$, is defined as

$$\hat{y}(x) = \sum_{i=1}^{m} W_i y_i \tag{2}$$

where

$$w_i = \frac{\exp\left(-\frac{||x-x_i||^2}{2\sigma^2}\right)}{\sum_{k=1}^{m} \exp\left(-\frac{||x-x_k||^2}{2\sigma^2}\right)}. \tag{3}$$

GRNN has one tunable parameter, namely, spread factor [in (2) above]. When the spread factor is small, the RBF is steep, and only a small number of training samples closest to the input contribute to the network output, which results in a rough response surface. As the spread factor increases, the RBF becomes wider and many more training samples contribute to the network output, which leads to a smoother response surface. Therefore, spread factor dictates GRNNs prediction performance.

As discussed in Section II, MLP networks have several shortcomings that make design, particularly automatic design, of MLPs a difficult task [1], [8]. GRNNs, on the other hand, have several advantages, including: 1) it has one design parameter (i.e., spread factor); 2) it is easy to train since it is a one-pass algorithm; 3) it can accurately approximate functions from sparse and noisy data; and 4) it can converge to the conditional mean surface by increasing the number of data samples [39]. It is these unique advantages that make us to choose GRNN in our automatic ANN modeling scheme.

In spite of having several unique properties that are desirable for TSF, GRNNs have not been as popular as other networks for TSF. A few of recent studies on GRNN for TSF include [27], where a comparison study was performed on using different machine-learning models, including GRNN, for TSF. In their study, Ahmed *et al.* [27] used the basic form of GRNN and used 10-fold cross-validation to select the best spread factor out of a set of spread factors ranging from 0.05 to 0.7. They concluded that GRNN is inferior over MLP and GP, but is generally more robust cross-difference series. Some recent applications of GRNNs include financial TSF [41], river flow forecasting [25], load forecasting [42], [43], and plasma etching prediction [44].

IV. Proposed Methodology

Our primary goal in this paper is to develop an ANN modeling scheme for TSF, which is as automated as possible so that it can be used to effectively and efficiently model a large number of time series. More specifically, we have devoted our effort toward the following three objectives in designing our modeling scheme: 1) it requires minimal human intervention; 2) it is computationally efficient for a large number of series; and 3) it has good overall forecasting performance. Our proposed automatic ANN modeling scheme consists of two essential components, namely, *preprocessing* and *ANN modeling*, which are described in detail in the following two subsections.

A. Preprocessing

A real-world time series, regardless of its type and its application, often contains various patterns, such as, trend, seasonality, and outlier. It is well recognized that the treatment of these patterns, which we call "preprocessing" in this paper, is a critical part of time-series modeling [45, p. 29]. Preprocessing time series typically includes two steps, namely, pattern identification and pattern treatment. Manually identifying time-series patterns via visual analysis has been popularly used in TSF, but it is labor intensive and is limited to a small number of series. When the number of series is large, an automatic pattern identification process is necessary, which is often a challenging task. A few studies, for example [1], have already looked at the pattern identification problem. Toward automatic ANN modeling, our preprocessing in this paper focuses on automatic identification and treatment of three time-series patterns, namely, outliers, trend, and seasonality. Normalization required for neural network modeling in general is also included in our preprocessing.

1) Outlier Identification and Treatment: Outliers are isolated observations that are significantly different in value from the pattern in the rest of the series. Adya *et al.* [46] used large second difference value as an indication of outliers. Based on our own previous experience, in this paper, we define an outlier as a point whose absolute value is four times greater than the absolute medians of the three consecutive points before and after, respectively, of the point. That is, x_i is an outlier if its value satisfies the following condition:

$$|x_i| \geq 4 \times \max\{|m_a|, |m_b|\} \tag{4}$$

where $m_a = \text{median}(x_{i-3}, x_{i-2}, x_{i-1})$ and $m_b = \text{median}(x_{i+1}, x_{i+2}, x_{i+3})$.

When an observation is deemed to be an outlier based on (4), its value is simply replaced with the average value of the two points that are immediately before and after the outlier.

2) Trend Identification and Treatment: Whether or not neural networks can effectively model trend in time series is an open question. The recent work by Qi and Zhang [47] was dedicated to answering this specific question. Their conclusion was that for most nonlinear and/or stochastic time series, detrending data by taking the first difference is the best practical approach for building effective ANN forecasting models. Their study was based on MLP networks. For GRNNs, however, there has been no such study performed exclusively. Since GRNNs perform prediction based on the similarity of the input point to the historical data points in the input space (see Section III for descriptions of GRNNs), GRNNs are inherently ineffective in modeling trend. Thus effective detrending, especially removing global trend, is more important for GRNNs than other ANN models.

There are different types of trend associated with time series, including global linear, local linear, nonlinear, and stochastic trend [45, p. 15]. Common approaches for handling trend include curve fitting, filtering, and differencing [45, p. 16]. A big challenge is on determining whether a series has trend and, if it does, what type of trend, that is, linear, nonlinear, or stochastic it is. Even more challenging is to do this in a systematical, not manual, fashion. Aiming at an automatic modeling scheme, in this paper, we propose a generic detrending scheme—subtracting full-season means. Under this detrending scheme, a series is first split into segments. The length of the segments is equal to the length of seasonality, that is, 12 for monthly series and 4 for quarterly series. The mean of the historical observations within each of these segments is subtracted from every historical observation in the segment. Let $\{m_1, m_2, \ldots, m_k\}$ be the means of the k segments of a series and l be the length of seasonality. The detrended series, dt_i, is related to the original series, x_i, as follows:

$$dt_i = x_i - m_{j=\text{cell}\left(\frac{i}{j}\right)}, \quad i = 1, 2, \ldots, n \tag{5}$$

where n is the length of the series. Such a detrending scheme is effective for removing global trends (both linear and nonlinear). While forecasting models can be effectively built on the detrended series, global trend (represented in full-season means) needs to be added back to the model outputs to obtain future forecasts. While the full-season means for historical observations can be easily calculated as shown above, the means for future forecasts are unknown. How can we find means for future forecasts? There are several ways to estimate the season means of future points. For example, we can create a prediction model to estimate future means based on the historical means, if the series is long enough and thus the number of means is large enough. In this paper, we adopt different strategies for short and long series, respectively. For short series ($n < 60$ points), we take the future means as the same as the last historical means. For long series, the future means are estimated based on the average of the last two historical means. Effectiveness of our detrending scheme will be discussed in Section VI by comparing to other detrending schemes.

3) Seasonality Identification and Treatment: For addressing seasonality, in literature there are generally two different ANN modeling strategies, that is, direct and deseasonalized. Opinions regarding these two strategies are mixed. Nelson *et al.* [9] and more recently Zhang and Kline [10] found that networks trained on deseasonalized data performed significantly better than those trained on raw data, while Heravi *et al.* [6] were in favor of modeling neural networks directly on raw data since they had a concern on potential nonlinearity induced by the seasonality adjustment.

There are several ways to identify seasonality of a time series. Most of them are manual processes. One example of such manual processes is by interpreting the correlogram—a graph in which the sample autocorrelation coefficients are plotted against the different lags. Automatically determining the seasonality of a series is not a trivial task. Zhang and Kline [10] used a simple rule of thumb for identifying seasonality. That is, a series is seasonal if the autocorrelation coefficient at the first seasonal lag is greater than $2/\sqrt{2}$, where n is the length of the series. During our preliminary study, this simple rule of thumb was applied to the NN3 competition data. It worked reasonably well for the short seasonal series, but misclassified the majority of the long nonseasonal series. In this paper, we adopt the following seasonality identification strategies for short and long series, respectively.

A short series ($n \leq 60$) is

$$\begin{cases} \text{seasonal}, & \text{if } \rho(1) > \frac{2}{\sqrt{n}} \\ \text{non} - \text{seasonal}, & \text{otherwise.} \end{cases} \tag{6}$$

A long series ($n > 60$) is

$$\begin{cases} \text{seasonal}, & \text{if both } \rho(1) \& \rho(2) > \frac{2}{\sqrt{n}} \\ \text{non} - \text{seasonal}, & \text{otherwise} \end{cases} \tag{7}$$

where n is the length of the series, and $\rho(1)$ and $\rho(2)$ are the autocorrelation coefficients at one and two seasonal lags, respectively. Using our seasonality criterion on the NN3 competition data, we correctly identified 45 out of all 54 seasonal series and 53 out of all 57 nonseasonal series.

For seasonal series, a simple deseasonalization procedure [8] is adopted. It performs deseasonalizing by subtracting the seasonal average. Let l be the length of seasonality (12 for monthly data and 4 for quarterly data) and $a_k, k = 1, 2, \ldots, l$

be the seasonal averages. The deseasonalized series is

$$ds_i = x_i - a_{j=\text{mod}(i,l)+1}, i = 1, 2, \ldots, n. \quad (8)$$

To perform forecasting, we simply add the seasonal average back to the outputs of the model that is built on the deseasonalized series. The effectiveness of deseasonality to GRNNs is discussed in Section VI.

4) Normalization: After performing the above-mentioned pattern treatment, data points are then normalized to a range of [0, 1] using linear scaling based on the minimal and maximum values of the series. Other normalization methods can also be used. Sensitivity of our modeling scheme to different normalization methods will be discussed in Section VI.

B. ANN Modeling

ANN modeling in this paper involves the following three efforts: 1) determining the spread factor of GRNN; 2) determining the inputs of GRNN; and 3) deciding on MSA forecasting strategies, all of which are described in detail in the subsequent subsections.

1) GRNNs Spread Factor: As discussed in Section III, spread factor of GRNNs is the only design parameter that directly affects prediction performance of GRNNs. Unfortunately, there is no single spread factor that works well for all GRNN applications. Moreover, there is no good analytical method that allows for accurately determining the best spread factor for a given application. Haykin [38, p. 299] provided a guidance on determining spread factor, that is, $\sigma = d_{\max}/\sqrt{2n}$, where $d_{\max}$ is the maximum distance between the training points, and n is the number of training points. The problem with this guidance is that the spread factor is dependent on the number of training samples. That is, it tends to yield big spread factor for short series and small spread factor for long series. To date, empirically determining the spread factor by trial and error is still the common practice in the design of GRNNs. However, an empirical approach is time consuming especially when the number of series is large. One could use an optimizer (e.g., genetic algorithm) as a wrapper to determine the optimal spread factor for each individual time series by minimizing an objective function (e.g., errors on the holdout samples). However, such optimization is also computationally expensive when the number of time series is large. Moreover, such optimization may only work well when the number of observations is large, that is, for long series. For short series, the spread factor that is optimally selected based on a small number of training samples may not generalize well to out-of-sample forecasting. Instead of searching for a single spread factor for each individual series, this paper tackles the spread factor issue by using multiple (e.g., three in this paper) GRNNs, each of which uses one spread factor from a spread factor set. The spread factor set is series dependent and is related to the distribution of the training samples of the series in the input space. As discussed in Section III, GRNN is an instance-based model, that is, it performs prediction based on the similarity of the input point to the training sample points in the input space. We hypothesize that the spread factor of the GRNN model should be related to how the training samples are distributed in the input space. Thus in this paper we propose the spread factor set to be $\{d_{50}, d_{75}, d_{95}\}$, the 50th, 75th, and 95th percentiles of the nearest distances of all training samples to the rest of the points. The three spread factors are calculated specifically for each individual series. In Section VI, we will explore how sensitive the number of GRNNs is to our model performance.

All three GRNNs take the same input, and the outputs of the three GRNNs are combined to arrive at the final prediction.

Our strategy of using multiple GRNNs with different spread factors in our modeling scheme is inspired by the success of model combination in the domain of machine learning. A large body of literature has shown that model combination can improve model performance for both classification and regression tasks [48]. Even for time series, several studies (e.g., [13], [32], [33]) have shown that combining forecasts can be an effective tool for improving forecasting accuracy. To the best of our knowledge, however, using model combination for alleviating the difficulty of individually determining the spread factors of GRNNs in modeling a large number of series has not been studied.

In the machine learning literature, one can find many model combination methods derived from different underlying frameworks [49], ranging from Bayesian probability theory [50], to fuzzy sets [51], to Dempster–Shafer evidence theory [52], to group decision-making theory (e.g., majority voting, weighted majority voting, and Borda count). Kuncheva [53] provided a good overview of model combination methods. In the field of time series forecasting, simple combination methods (e.g., simple average) have been popularly used, since studies have shown that a simple average often outperforms the more sophistic combination methods [33]. Recently, Jose and Winkler [54] proposed two new, simple combination methods—the trimmed and the Winsorized means. While one may choose to empirically select the best combination method for the given problem, for our automatic modeling scheme, we use the simple average as the combination method. The effect of using different combination methods to the forecasting performance of our modeling scheme is investigated through sensitivity study in Section VI.

2) GRNN Inputs: Inputs to GRNNs are typically the lagged observations of a time series. Several studies, for example, [1], have indicated that selecting model inputs is an important, if not the most important, task in TSF modeling. In machine learning, input selection (or feature selection) is a research topic on its own. Dash and Liu [55] have carried out a comprehensive overview of feature selection techniques. Broadly speaking, feature selection methods can be categorized into filter and wrapper approaches [56]. The filter approach selects features as a result of preprocessing based on the properties of the data itself, independent of the learning algorithm. The wrapper approach, on the other hand, uses the learning algorithm as part of the evaluation. Wrapper approach generally gives a feature set that has better performance, but is also computationally more expensive.

Algorithm 1 Pseudocode for Performing the Proposed ANN Modeling

FOR each time series {
// *1. Preprocessing*
1.1 Perform outlier identification and treatment if applicable
1.2 Perform trend identification and detrending if applicable
1.3 Perform seasonality identification and deseasonalizing if applicable
// *2. ANN modeling*
2.1 Hold out the last 18 historical observations for validation and keep the remaining samples for training
For each of the input lags of [1,2,…,12] {
2.2 Determine spread factor set for GRNNs
2.3 Train each GRNN on the training set
2.4 Test each GRNN on the hold out samples
2.5 Fuse outputs of all GRNNs to obtain initial forecasts
2.6 Add seasonality back, if applicable
2.7 Add trend back, if applicable
2.8 Calculate sMAPE
2.8 Record the best input lag and corresponding spread factor set }
2.9 Use all historical observations to retrain GRNNs with the best input lag and the spread factor set
2.10 Perform out-of-sample forecasting
2.11 Add seasonality back if applicable
2.12 Add trend back if applicable
2.13 Calculate the forecasting sMAPE
}
END

Studies specifically on feature selection for TSF are sparse. Huang *et al.* [57], Tikka *et al.* [58], and Crone and Nikolopoulos [59] are a few sample studies on this topic. The majority of real-world applications still rely on experiments and/or empirical intuition for determining the model inputs. One can simply try different inputs and pick one with the smallest error (or other performance measure) from the holdout testing set. In this paper, we adopt such an experimental approach for determining the inputs to our GRNNs. To minimize the search time, we consider only the contiguous lags as inputs and we further limit the maximum lag to one full season (12 for monthly data). For short series ($n \leq 60$), we simply pick a fixed lag equal to the season length (12 for monthly data) considering the fact the empirically selected input based on a small number of training samples may not generalize well to out-of-sample forecasting.

3) Multiple-Step-Ahead Forecasting: Depending on the length of forecasting horizon (the number of points into future), time series can be single-step-ahead (SSA) or mult-step-ahead (MSA). SSA forecasting has the shortest forecasting horizon and is relatively easy, thus it has been widely discussed in the literature. MSA forecasting, however, has not been widely studied despite the fact that it is often of greater value in many real-world applications. MSA forecasting is much more challenging because it involves dealing with growing uncertainties caused by various sources (e.g., error accumulation and lack of information) [60].

For MSA forecasting, there are generally two different modeling approaches, that is, direct and recursive. The direct approach uses multiple prediction models, each of which is explicit for one of the k steps ahead. The recursive approach performs one-step-ahead predictions recursively—using the prior prediction as an input on the second and subsequent steps—until it reaches the desired prediction horizon. The recursive approach is simple and intuitive, but has a major drawback of prediction error accumulation. The direct approach, on the other hand, is computationally more expensive since it requires training multiple models. The relative performance of the two modeling approaches generally depends on the time series and on the prediction model used. For neural network models, several studies, for example, [60], have shown that the direct approach is more accurate than the recursive approach in MSA forecasting. Atiya *et al.* [8] even analytically proved the superiority of the direct approach over the recursive approach. Therefore, this paper employs the direct approach for MSA forecasting. We will discuss the difference between the two MSA approaches in terms of forecasting performance in Section VI.

A step-by-step procedure for building the ANN models for TSF using the proposed modeling scheme is summarized in Algorithm 1.

V. Experimental Results

A. NN3 Competition Datasets

In this paper, the NN3 time-series competition dataset [12] is used for validating the ANN modeling scheme proposed.

The NN3 competition has two datasets. Dataset A is a complete dataset consisting of 111 monthly time series drawn from a homogeneous population of empirical business time series. Dataset B, on the other hand, is a small subset of Dataset A, which consists of 11 time series. The complete dataset (Dataset A) contains a balanced sample of both long (more than 100 observations per series) and short (less than 50 observations per series) time series, as well as an even split of seasonal and nonseasonal time series. That is, it includes 25 short-seasonal, 25 long-seasonal, 25 short-nonseasonal, 25 long-nonseasonal, and the 11 time series of the reduced dataset (Dataset B). In this paper, we use all 111 series (the complete dataset) for our experimental study. Given the historical observations of each time series, the objective of the competition is to forecast the future 18 values, that is, to forecast x_{t+h}, where $h = 1, 2, \ldots, 18$, based on the given historical observations of $x_1, x_2, \ldots, x_t$.

B. Performance Measures

For evaluating forecast accuracy, several different measures have been proposed in the literature. Some of them are based on percentage errors and others are based on relative errors. Hyndman and Koehler [61] conducted a comparison study on different measures of forecast accuracy. For model evaluation and determining the winners of the competition, the NN3 competition organizers chose the symmetric mean absolute

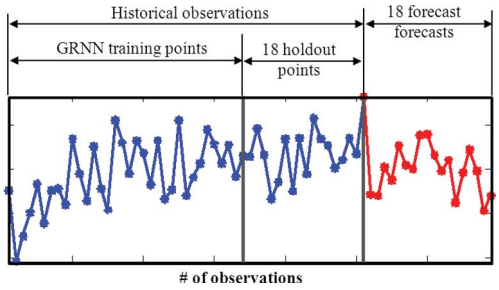

Fig. 2. Typical series segmentation for model training, testing, and future forecasting.

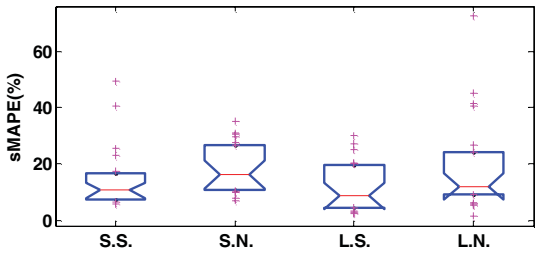

Fig. 3. Boxplots of sMAPEs for groupwise series. The four groups are short seasonal (S.S.), short nonseasonal (S.N.), long seasonal (L.S.), and long nonseasonal (L.N.) (*Note:* for all boxplots shown in this paper, red line at the notch of the box indicates median, two ends of the box represent 25th and 75th percentiles, respectively, and "+" represents points outside 25th and 75th percentiles).

percentage error (sMAPE) defined in (9) below

$$sMAPE = \frac{1}{M}\sum_{i=1}^{M}\frac{|Y_i| - |\hat{Y}_i|}{(|Y_i| + |\hat{Y}_i|)/2} \times 100 \qquad (9)$$

where Y_i and $\hat{Y}_i$ are the true and predicted values, respectively, at ith time point, and M is the number of forecasting points. sMAPE is a relative error measure, which allows for combining errors computed for different series into one number. For each time series, the sMAPE is computed over the 18 forecasts, that is, $M = 18$ in (9). Such per-series sMAPE is then averaged over all 111 series to obtain the so-called overall average sMAPE.

C. Modeling Details

Using direct approach for multi-step-ahead forecasting of the NN3 competition data, 18 models are used for each series. Each of the 18 models performs i-step-ahead forecasting, where $i = 1, 2, \ldots, 18$. It is worth pointing out that each of the 18 models consists of three GRNNs with different spread factors as described in Section IV.

To empirically determine the number of GRNN inputs, for each of the 111 series, the last 18 *historical* observations are reserved as holdout testing samples and the remaining portion of the historical observations are used for training the GRNNs (as shown in Fig. 2). Since all of the 111 series considered are monthly series, the seasonal length here is 12. For the long series ($n > 60$), k lagged inputs with $k = 1, 2, \ldots, 12$ is tried as the GRNN inputs, and the one with the smallest sMAPE on the holdout testing samples is chosen as the final input for the series. After determining the number of lagged inputs, the networks are retrained over all historical observations (including the holdout samples), and such trained networks are used for out-of-sample forecasting. For the short series ($n \leq 60$), on the other hand, a fixed 12-lagged input is used. It is worth noting that the number of lagged inputs is kept the same for all 18-horizon models.

The ANN modeling scheme proposed in this paper is implemented using MATLAB (R2009b, The Mathworks, Inc., Natick, MA, USA), while GRNNs are implemented using the Neural Network Toolbox of MATLAB.

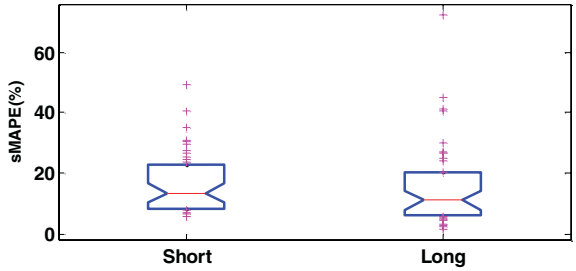

Fig. 4. Boxplots of sMAPEs—short series versus long series.

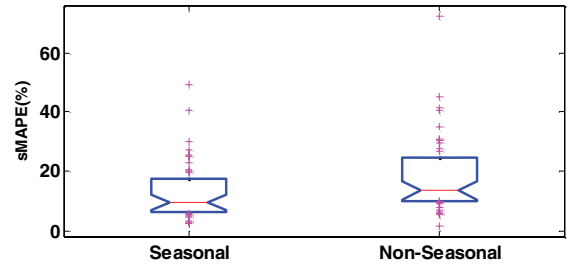

Fig. 5. Boxplot of sMAPEs—seasonal series versus nonseasonal series.

D. Results

To assess the performance of our automatic modeling scheme, we employ a fixed-origin evaluation of the 18 forecasts, which is in accordance to the NN3 competition. Applying our modeling scheme to forecast the future 18 points for each individual series, we obtain the per-series sMAPEs based on the 18 future points. The overall average sMAPE of the forecasts for all 111 series is 15.80%.

To assess our model performance over different segments of the time series, we calculate the descriptive statistics of the sMAPEs individually corresponding to the four series segments and show them in boxplots in Fig. 3. We then specifically compare the boxplots between 50 short series and 50 long series in Fig. 4, and between 50 seasonal and 50 nonseasonal series in Fig. 5. From those boxplots one can see that our model generally performs better for seasonal series than nonseasonal series, and our model performs slightly better for long series than short series. The less favorable performance of our model on nonseasonal series indicates that nonseasonal series, especially short nonseasonal series, is more difficult to model.

We also calculate and plot out in Fig. 6 the sMAPEs over six levels of forecasting horizons. It shows that generally the forecasting error increases as the forecasting horizon increases, which is in accordance with intuition.

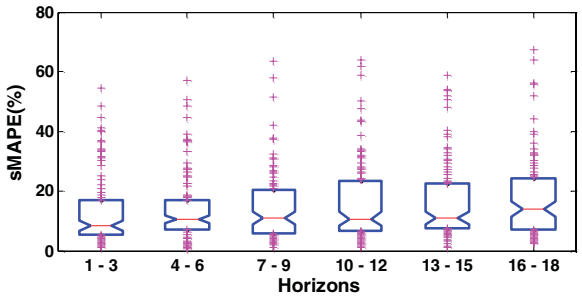

Fig. 6. Boxplots—sMAPE over different forecast horizons.

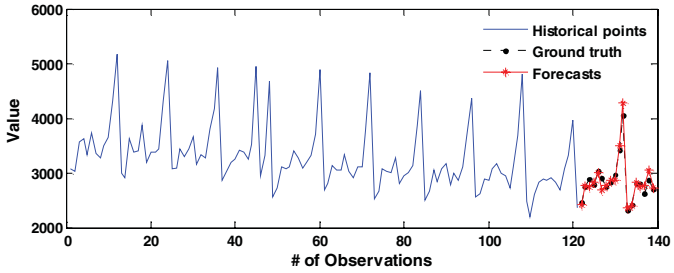

Fig. 7. Historical observation and future forecasts for Series 68 for which our model yields an sMAPE of 2.7% (the best series).

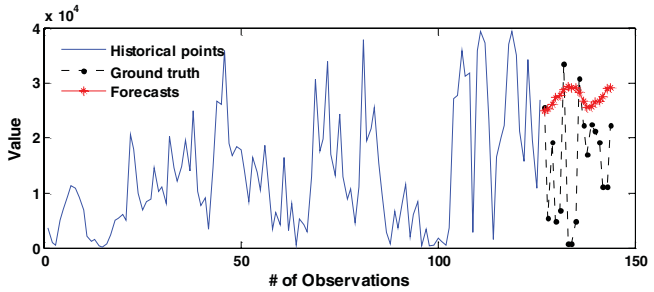

Fig. 8. Historical observation and future forecasts for Series 93 for which our model yields an sMAPE of 73% (the worst series).

Out of all 111 series, our per-series sMAPEs are generally below 40%, with the best sMAPE being approximately 2.7% (see Fig. 7). However, several of the series have much higher sMAPE. In particular, Series 93 has the sMAPE close to 73%. Fig. 8 shows the historical observations, the true future 18 points, and the 18 forecasts predicted by our model, respectively. Clearly this specific series is a difficult one and our automatic model scheme could not capture the underlying pattern well.

In Table I, the overall average sMAPE of our model is also compared to those of the other top-performed models submitted to the NN3 competition, where the model IDs with letter "C" as prefix stand for computational intelligence (CI) models and the model IDs with letter "B" stand for statistical benchmark models. Table I was compiled from [12]. From Table I one can see that our automatic modeling scheme proposed in this paper outperforms almost all of the submitted CI models except Model C27—Echo state networks. Our model performance is also comparable to the top-performed statistical benchmark models. For details of these submitted and benchmark models, refer to [12]. Had we somehow

TABLE I
PERFORMANCE COMPARISON BETWEEN OUR MODEL AND TOP-PERFORMED MODELS SUBMITTED TO THE COMPETITION

ID	Method	sMAPE	Rank
B09	Stat. Benchmark	14.84%	1
B07	Stat. Benchmark	14.89%	2
C27	Echo state networks	15.18%	3
B03	Stat. Benchmark - ForecastPro	15.44%	4
-	**Our proposed model**	**15.80%**	**5**
B16	DES	15.90%	6
B17	Comb S-H-D	15.93%	7
B05	Stat Benchmark - Autobox	15.95%	8
C03	Linear model + GA	16.31%	9
B14	SES	16.42%	10
B15	HES	16.49%	11
C46	Regression tree ensemble	16.55%	12
C13	kNN	16.57%	13

Note: In our original submission to the NN3 Competition, we used series-independent design, that is, the spread factors used for GRNNs are the same for all 111 series, which led to a higher sMAPE of 18.58%. Since then we have adopted a series-dependent design that is reported in this paper, which resulted in a much improved forecasting performance.

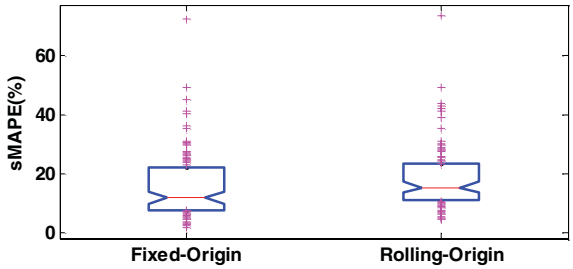

Fig. 9. Boxplots—comparison of sMAPEs between fixed origin and rolling-origin evaluations.

reduced the sMAPE of Series 93 from 73% to 20% (assuming we can achieve that by fine-tuning the design parameters of our automatic ANN model), our overall sMAPE would have been around 15.30%, which would put our model further close to the best CI model (C27).

To desensitize the error measures to special events at any single origin, we also assessed our model performance based on the rolling-origin evaluation [62]. We did six origins, that is, $T, T+1, \ldots, T+5$, where T is the last historical observation. The sMAPEs of these six origins are then averaged for each series. The overall average sMAPE of the 111 series is 17.94% (as opposed to 15.80% for fixed-origin). The boxplots for both fixed-origin and rolling-origin evaluations are compared in Fig. 9.

VI. DISCUSSION

As discussed in detail in Section IV, our modeling scheme involves adopting several important design strategies, including subtracting full-season means for detrending, seasonal average subtraction for deseasonalizing, and fusion of multiple GRNNs with different spread factors for simplifying the determination of single spread factor. These design strategies are introduced mainly for automating ANN modeling. By

applying our modeling scheme to the NN3 competition data, we have demonstrated that those design strategies adopted in our modeling scheme are reasonably effective in TSF. In this section, we would like to have a preliminary investigation on how those individual design strategies affect our forecasting performance and the sensitivity of some design choices to the model performance. To that end, both contrast analysis and sensitivity analysis are conducted in the following two subsections. Limitations and the potential use of our modeling scheme are also discussed in this section.

A. Contrast Analysis

For contrast analysis, we perform several comparisons between different designs, attempting to answer the following questions.

1) How effective is our detrending strategy?
2) Is deseasonalization a necessary preprocess step for GRNNs?
3) How effective is using multi-GRNN fusion for tackling the spread factor issue in GRNN modeling?
4) How different is it between the two MSA approaches—direct and recursive?

Specifically, the following six comparisons are conducted. To assess the effectiveness of the detrending strategy adopted in our modeling scheme, which is described in detail in Section IV-A, we compare it against the first-difference detrending that is commonly used in TSF. We designate this first comparison as "auto versus first-difference detrending (D1)." We also compare our design against no detrending, which is designated as "auto versus no detrending (D2)."

To understand if deseasonalization improves GRNNs' performance, we compare our model that uses seasonal average subtraction for deseasonalizing against the design that does not use deseasonalization. We call this comparison "auto versus nondeseasonalizing (D3)."

To assess the effectiveness of our strategy of addressing the spread factor issue, which involves fusion of three GRNNs with different spread factors, we compare our design against the following two designs. The first design uses a single spread factor that is estimated using Haykin's formula, that is, $\sigma = d_{\max}/\sqrt{2n}$, where $d_{\max}$ is the maximum distance between the training samples, and n is the number of training samples. The second design involves finding the best single spread factor for each series by grid searching. The search space for the spread factor issue ranges from 0.1 to 1.2 with an increment of 0.2. We designate these two comparisons as "auto versus single calculated SF (D4)" and "auto versus single optimal SF (D5)," respectively.

The last contrast analysis we conducted is on comparing the two MSA design strategies—direct and recursive (see Section IV-B). Since our automatic modeling scheme uses direct approach, we designate this comparison as "auto versus recursive MSA (D6)."

In all of these six comparisons, the design configuration (i.e., the full design parameters) used in our modeling scheme is treated as the baseline. For each of these comparisons, the design to be compared against is the result of changing one subset of design parameters corresponding to the specific design, while keeping the rest of the design parameters unchanged.

TABLE II
WILCOXON SIGNED-RANK TEST RESULTS FOR ALL COMPARISON PAIRS

	Comparison	sMAPEs (%)	Z-value	p-value
1	Auto versus first-difference detrending (D1)	18.51	−3.143	0.0017
2	Auto versus no detrending (D2)	16.68	−2.071	0.0383
3	Auto versus nondeseasonalizing (D3)	17.74	−4.836	< 0.0001
4	Auto versus single calculated SP (D4)	16.37	−3.013	0.0026
5	Auto versus single optimal SP (D5)	15.83	−0.118	0.9057
6	Auto/direct versus recursive MSA (D6)	16.05	−1.351	0.1769

To obtain statistical significance of the differences for all of the pairwise comparisons, we use Wilcoxon signed-rank test (WSRT) [63]. WSRT is a nonparametric hypothesis test for the median difference. Compared to the t-test that requires normal distribution of the samples, WSRT makes fewer and less stringent assumptions on the sample distributions and thus is more powerful in detecting the existence of significant differences [64]. In our WSRT setting, the matched pair samples are the per-series sMAPEs of the 111 time series, which are calculated using the 18 future forecasts.

Table II summarizes the test results, from which the following conclusions can be made.

1) The detrending scheme used in our model is statistically significantly different (with a p-value of 0.0017) from the popularly used first-difference detrending method. More specifically, our approach yields a smaller overall sMAPE than the first-difference detrending does since the z-value is negative, our detrending design is also statistically significantly better than no trending design (with a z-value of −2.071 and a p-value of 0.0383).
2) Deseasonalizing makes significant improvement (with a z-value of −4.836 and a p-value less than 0.0001), which indicates that deseasonalization is important for GRNN models.
3) The fourth-comparison results show that our spread factor determination strategy is significantly better than using single spread factor calculated using the Haykin's formula (with a z-value of −3.013 and a p-value of 0.0026).
4) With a p-value of 0.9057 that is much greater than 0.005 (D5), we are confident that our spread factor determination strategy is not statistically significantly different (in other words, they are equally effective) in terms of the sMAPE from using search to find the spread factor. Our strategy, however, is computationally advantageous since it only involves training and evaluating three GRNNs as opposed to many more GRNNs in the search method.

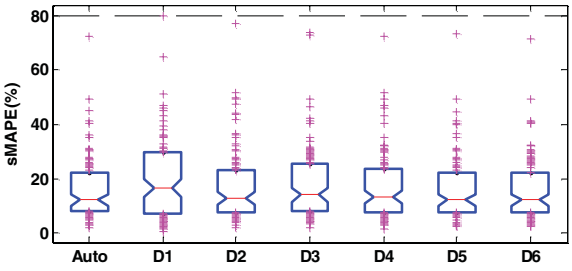

Fig. 10. Boxplots—comparison of sMAPEs of our modeling scheme against those from the six designs shown in Table II.

TABLE III
ANOVA RESULTS

Source	DF	Sum of Squares	Mean Square	F-value	p-value
Normalization methods	2	3.8×10^{-6}	1.9×10^{-6}	0.44	0.653
Number of GRNNs	1	7.9×10^{-6}	7.9×10^{-6}	1.83	0.201
Fusion methods	2	1.5×10^{-5}	7.7×10^{-6}	1.78	0.210

Note: DF = degree of freedom.

5) Even though the direct approach we adopted as our MSA strategy shows a slightly better performance than the recursive approach in terms of sMAPE (D6 in Table II), the difference between the two approaches is statistically insignificant with a p-value of 0.1769.

Fig. 10 summarizes all of the six comparisons in boxplots of sMAPEs, where "auto" refers to our automatic modeling scheme and D1 thru D6 are for the six designs we compared in Table II.

We would like to point out that the six comparisons conducted here do not cover all possible pairwise combinations of all designs. Also these comparisons only assess the independent effect, but not the interaction among the designs. Therefore, we have to be cautious in drawing general conclusions from this paper.

B. Sensitivity Analysis

To find out how sensitive the design parameters of our automatic neural network modeling scheme are to the forecasting accuracy, a design of experiments (DOE) study is performed. The three factors or design parameters considered are: 1) normalization methods; 2) the number of GRNNs for fusion; and 3) fusion methods. The number of levels for each of the three factors is set as follows: three levels for normalization methods, that is, min–max linear normalization to [0 1], min–max linear normalization to [−1 1], and mean and variance normalization; two levels for the number of GRNNs, that is, three and five, where the five spread factors for five GRNNs are 5th, 25th, 50th, 75th, and 95th percentiles, respectively, of the nearest distances of all training samples to the rest of samples; and three levels for fusion methods, that is, mean, median, and trimmed mean [54]. A full factorial DOE design results in a total of 18 experiments. Analysis of variance (ANOVA) was performed on the sMAPEs of the 18 experiments. Table III shows the ANOVA results.

From Table III one can see that with p-value greater than 0.05, our model is generally not sensitive to the three design factors considered, that is, normalization, the number of GRNNs, or fusion methods. Once again, we have to be cautious about generalizing our conclusions here until a more comprehensive DOE study is completed.

C. Limitations

The main limitation of the modeling scheme proposed in this paper is that, since it is designed for automatically modeling large-scale time series, performance of our model may not always be superior for certain time series over other models that are carefully handcrafted specifically for the series. Also, not all of the design parameters in our model are optimally chosen, but rather designed in ad hoc fashion. It would be good to extend our modeling scheme by including an optimizer as a wrapper, so that the majority of the design parameters can be automatically optimized based on the characteristics of the time series. In particular, we have not exhaustively explored the preprocessing strategies for better handling trend, seasonality, and other nonstationary features. However, our modeling scheme is flexible enough to incorporate those preprocessing strategies. In addition, our modeling scheme has only been validated on the NN3 competition data, which consists of various monthly, business-type time series. Hence the conclusions drawn in this paper may not generalize well over nonbusiness time series. It would be interesting to find out whether or not our modeling scheme works equally well for nonbusiness-type series.

We envision that our automatic ANN modeling scheme proposed here can potentially benefit the TSF community in the following three aspects:

1) as a tool for large-scale TSF research, where a large number of series are involved;
2) as an initial model for users to gain some general knowledge of the series, when dealing with single or a small number of series;
3) as a default setting in forecasting tools, where users have options of choosing different modeling strategies or design configurations.

VII. CONCLUSION

ANNs have been actively used for various TSF applications for decades. However, we are still lacking a systematic ANN modeling process and strategy. This paper attempted to develop an automatic ANN modeling scheme that is based on a special type of network, GRNN. By introducing several design strategies, we were able to make our automatic modeling scheme effective for TSF. The reasonably good performance of our model on the NN3 competition data that consisted of heterogeneous forecasting problems validated the applicability and the effectiveness of our model for large-scale TSF. The following are the main conclusions drawn in this paper.

1) GRNN is potentially a good candidate for automatic ANN modeling for TSF, thanks to several of its unique properties (e.g., single design parameter).

2) Our automatic ANN modeling scheme that has GRNN as its base model is largely robust with respect to most of the design parameters, which makes it feasible for automatically modeling large-scale time series.
3) Computationally, our modeling scheme is reasonably efficient because of the property of fast learning of GRNNs and the design strategies carefully adopted in this paper.

Automating ANN modeling is certainly a challenging task and has a long way to go before maturing. As an initial effort toward automatic ANN modeling, we employed relatively simple strategies in both preprocessing and ANN modeling steps. Many improvements can be made to our modeling scheme to make it much more effective and efficient. For example, we can extend out input selection to include more comprehensive feature selection methods discussed in Section IV-B for identifying more salient inputs (both contiguous and noncontiguous lags). As stated earlier, a more thorough sensitive analysis via DOE to assess the robustness of our modeling scheme to different design configurations is certainly worth pursuing. Our future work may also include applying the modeling scheme to other large-scale datasets, for example, the M3 competition data.

ACKNOWLEDGMENT

The author would like to thank the NN3 Competition organizers for allowing use of the data in this paper. The author would also like to thank the anonymous reviewers for their constructive comments.

REFERENCES

[1] G. P. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecast.*, vol. 14, pp. 35–62, Mar. 1998.
[2] S. Crone and P. Grafeille, "An evaluation framework for publications on artificial neural networks in sales forecasting," in *Proc. Int. Conf. Artif. Intell.*, Las Vegas, NV, Jun. 2004, pp. 221–227.
[3] T. Hill, M. O'Connor, and W. Remus, "Neural network models for times series forecasting," *Manage. Sci.*, vol. 42, no. 7, pp. 1082–1092, Jul. 1996.
[4] N. R. Swanson and H. White, "Forecasting economic time series using flexible versus fixed specification and linear versus nonlinear econometric models," *Int. J. Forecast.*, vol. 13, no. 4, pp. 439–461, 1997.
[5] W. Zhang, Q. Cao, and M. J. Schniederjans, "Neural network earning per share forecasting models: A comparative analysis of alternative methods," *Decision Sci.*, vol. 35, no. 2, pp. 205–237, 2004.
[6] S. Heravi, D. R. Osborn, and C. R. Birchenhall, "Linear versus neural network forecasts for European industrial production series," *Int. J. Forecast.*, vol. 20, no. 3, pp. 435–446, Jul.–Sep. 2004.
[7] L. J. Callen, C. C. Kwan, P. C. Yip, and Y. Yuan, "Neural network forecasting of quarterly accounting earnings," *Int. J. Forecast.*, vol. 12, no. 4, pp. 475–482, 1996.
[8] M. Adya and F. Collopy, "How effective are neural networks at forecasting and prediction? A review and evaluation," *Int. J. Forecast.*, vol. 17, nos. 5–6, pp. 481–495, Nov. 1998.
[9] M. Nelson, T. Hill, T. Renas, and M. O'Connor, "Time series forecasting using NNs: Should the data be deseasonalized first?" *J. Forecast.*, vol. 18, no. 5, pp. 359–367, 1999.
[10] G. P. Zhang and D. M. Kline, "Quarterly time series forecasting with neural networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1800–1814, Jun. 2007.
[11] S. D. Balkin and J. K. Ord, "Automatic neural network modeling for univariate time series," *Int. J. Forecast.*, vol. 16, no. 4, pp. 509–515, 2000.
[12] S. F. Crone, M. Hibon, and K. Nikolopoulos, "Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction," *Int. J. Forecast.*, vol. 27, no. 3, pp. 635–660, 2011.
[13] S. Makridakis and M. Hibon, "The M3-competition: Results, conclusions, and implications," *Int. J. Forecast.*, vol. 16, no. 4, pp. 451–476, 2000.
[14] A. F. Atiya, S. M. El-Shoura, S. I. Shaheen, and M. S. El-Sherif, "A comparison between neural network forecasting techniques-case study: River flow forecasting," *IEEE Trans. Neural Netw.*, vol. 10, no. 2, pp. 402–409, Feb. 1999.
[15] D. Vere-Jones, "Forecasting earthquakes and earthquake risk," *Int. J. Forecast.*, vol. 11, no. 4, pp. 503–538, Dec. 1995.
[16] I. Maqsood, M. R. Khan, and A. Abraham, "An ensemble of neural networks for weather forecasting," *Neural Comput. Appl.*, vol. 13, no. 2, pp. 112–122, 2004.
[17] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Trans. Power Syst.*, vol. 16, no. 1, pp. 44–55, Feb. 2001.
[18] J. Farway and C. Chatfield, "Time series forecasting with neural networks: A comparative study using the airline data," *Appl. Stat.*, vol. 47, no. 2, pp. 231–250, 1995.
[19] J. Connors, D. Martin, and L. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 240–254, Mar. 1994.
[20] C. M. Kuan and T. Liu, "Forecasting exchange rates using feedforward and recurrent neural networks," *J. Appl. Econ.*, vol. 10, no. 4, pp. 347–364, 1995.
[21] C. L. Giles, S. Lawrence, and A. C. Tsoi, "Noisy time series prediction using a recurrent neural network and grammatical inference," *Mach. Learn.*, vol. 44, nos. 1–2, pp. 161–183, 2001.
[22] A. G. Parlos, O. T. Rais, and A. F. Atiya, "Multi-step-ahead prediction using dynamic recurrent neural networks," *Neural Netw.*, vol. 13, no. 7, pp. 765–786, Sep. 2000.
[23] X. B. Yan, Z. Wang, S. H. Yu, and Y. J. Li, "Time series forecasting with RBF neural network," in *Proc. IEEE Int. Conf. Mach. Learn. Cybern.*, vol. 8. Guangzhou, China, Aug. 2005, pp. 4680–4683.
[24] F. Liang, "Bayesian neural networks for nonlinear time series forecasting," *Stat. Comput.*, vol. 15, no. 1, pp. 13–29, Jan. 2005.
[25] M. Firat, "Comparison of artificial intelligence techniques for river flow forecasting," *Hydrol. Earth Syst. Sci.*, vol. 12, no. 1, pp. 123–139, 2008.
[26] Y. Bodyanskiy, I. Pliss, and O. Vynokurova, "Adaptive wavelet-neuro-fuzzy network in the forecasting and emulation tasks," *Int. J. Inf. Theories Appl.*, vol. 15, no. 1, pp. 47–55, 2008.
[27] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econ. Rev.*, vol. 29, nos. 5–6, pp. 594–621, 2010.
[28] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006.
[29] A. Sorjamaa, Y. Miche, R. Weiss, and A. Lendasse, "Long-term prediction of time-series using NNE-based projection and OP-ELM," in *Proc. IEEE World Congr. Comput. Intell.*, Jun. 2008, pp. 2674–2680.
[30] E. J. Bayro-Corrochano and N. Arana-Daniel, "Clifford support vector machines for classification, regression, and recurrence," *IEEE Trans. Neural Netw.*, vol. 21, no. 11, pp. 1731–1746, Nov. 2010.
[31] S. Hu, G. Dai, G. A. Worrell, Q. Dai, and H. Liang, "Causality analysis of neural connectivity: Critical examination of existing methods and advances of new methods," *IEEE Trans. Neural Netw.*, vol. 22, no. 6, pp. 829–844, Jun. 2011.
[32] J. S. Armstrong, "Combining forecasts," in *Principles of Forecasting: A Handbook for Researchers and Practitioners*, J. S. Armstrong Ed. Norwell, MA: Kluwer, 2001.
[33] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *Int. J. Forecast.*, vol. 5, no. 4, pp. 559–583, 1989.
[34] H. Zou and Y. H. Yang, "Combining time series models for forecasting," *Int. J. Forecast.*, vol. 20, no. 1, pp. 69–84, 2004.
[35] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Learn.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.
[36] C. P. Lim and W. Y. Goh, "The application of an ensemble of boosted Elman networks to time series prediction: A benchmark study," *Int. J. Comput. Intell.*, vol. 3, no. 2, pp. 119–126, 2006.
[37] M. Assaad, R. Bone, and H. Cardot, "A new boosting algorithm for improved time-series forecasting with recurrent neural networks," *Inf. Fusion*, vol. 9, no. 1, pp. 41–55, Jan. 2008.

[38] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1990.
[39] D. F. Specht, "A general regression neural network," *IEEE Trans. Neural Netw.*, vol. 2, no. 6, pp. 568–576, Nov. 1991.
[40] H. Husain, M. Khalid, and R. Yusof, "Automatic clustering of generalized regression neural network by similarity index based fuzzy C-means clustering," in *Proc. IEEE Region 10 Conf. (TENCON)*, vol. 2. Nov. 2004, pp. 302–305.
[41] W. M. Li, Y. Luo, Q. Zhu, J. W. Liu, and J. J. Le, "Applications of AR*-GRNN model for financial time series forecasting," *Neural Comput. Appl.*, vol. 17, nos. 5–6, pp. 441–448, 2008.
[42] D. X. Niu, H. Q. Wang, and Z. H. Gu, "Short-term load forecasting using general regression neural network," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 7. Guangzhou, China, Aug. 2005, pp. 4076–4082.
[43] M. M. Tripathi, K. G. Upadhyay, and S. N. Singh, "Short-term load forecasting using generalized regression and probabilistic neural networks in the electricity market," *Electr. J.*, vol. 21, no. 9, pp. 24–34, Nov. 2008.
[44] B. Kim, D. W. Lee, K. Y. Parka, S. R. Choi, and S. Choi, "Prediction of plasma etching using a randomized generalized regression neural network," *Vacuum*, vol. 76, no. 1, pp. 37–43, Oct. 2004.
[45] C. Chatfield, *The Analysis of Time Series: An Introduction*, 6th ed. Boston, MA: Chapman & Hall, 2004.
[46] M. Adya, F. Collopy, J. S. Armstrong, and M. Kennedy, "Automatic identification of time series features for rule-based forecasting," *Int. J. Forecast.*, vol. 17, no. 2, pp. 143–157, 2001.
[47] M. Qi and P. G. Zhang, "Trend time-series modeling and forecasting with neural networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 5, pp. 808–816, May 2008.
[48] B. Gavin, W. Jeremy, H. Rachel, and Y. Xin, "Diversity creation methods: A survey and categorization," *J. Inf. Fusion*, vol. 6, pp. 5–20, Dec. 2005.
[49] P. Bonissone, K. Goebel, and W. Yan, "Classifier fusion using triangular norms," in *Proc. 5th Int. Workshop Multiple Classifier Syst.*, Cagliari, Italy, Jun. 2004, pp. 154–163.
[50] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. Syst. Man Cybern.*, vol. 22, no. 3, pp. 418–435, May–Jun. 1992.
[51] S. Cho and J. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Trans. Syst. Man Cybern.*, vol. 25, no. 2, pp. 380–384, Feb. 1995.
[52] A. Al-Aniand and M. Deriche, "A new technique for combining multiple classifiers using the Dempster-Shafer theory of evidence," *J. Artif. Intell. Res.*, vol. 17, no. 1, pp. 333–361, Jul. 2002.
[53] L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. New York: Wiley, 2004.
[54] V. R. R. Jose and R. L. Winkler, "Simple robust averages of forecasts: Some empirical results," *Int. J. Forecast.*, vol. 24, no. 1, pp. 163–169, Jan.–Mar. 2008.
[55] M. Dash and H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, no. 3, pp. 131–156, 1997.
[56] R. Kohavi and H. J. George, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, nos. 1–2, pp. 273–324, Dec. 1997.
[57] W. Huang, Y. Nakamor, and S. Wang, "A general approach based on autocorrelation to determine input variables for neural networks for time series forecasting," *J. Syst. Sci. Complex.*, vol. 14, no. 3, pp. 297–305, 2004.
[58] J. Tikka, A. Lendasse, and J. Hollmen, "Input selection for long-term prediction of time series," in *Proc. 8th Int. Work-Conf. Artif. Neural Netw.*, 2005, pp. 1002–1009.
[59] S. F. Crone and K. Nikolopoulos, "Input variable selection for time series prediction with neural networks-an evaluation of visual, autocorrelation and spectral analysis for varying seasonality," in *Proc. 1st Eur. Symp. Time Series Predict.*, Helsinki, Finland, 2007, pp. 1–11.
[60] A. Sorjamaaa, H. Jin, N. Reyhania, Y. Jia, and A. Lendasse, "Methodology for long-term prediction of time series," *Neurocomputing*, vol. 70, nos. 16–18, pp. 2861–2869, Oct. 2007.
[61] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecast.*, vol. 22, no. 4, pp. 679–688, 2006.
[62] L. J. Tashman, "Out-of-sample tests of forecasting accuracy: An analysis and review," *Int. J. Forecast.*, vol. 16, no. 4, pp. 437–450, 2000.
[63] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, no. 6, pp. 80–83, Dec. 1945.
[64] M. Hollander and D. Wolfe, *Nonparametric Statistical Methods*. New York: Wiley, 1999.

Weizhong Yan (M'04–SM'08) received the Ph.D. degree in mechanical engineering from Rensselaer Polytechnic Institute, Troy, NY, in 2002.

He has been with General Electric Company, Fairfield, CT, since 1998. Currently, he is a Senior Researcher with the Machine Learning Laboratory, GE Global Research Center, Niskayuna, NY. His specialties include applying advanced data-driven analytic techniques to anomaly detection, diagnostics, and prognostics & health management of industrial assets, such as jet engines, gas turbines, and oil and gas equipment. He has authored over 70 publications in referred journals and conference proceedings and has filed over 25 U.S. patents. His current research interests include large-scale analytics, ensemble learning, and time series forecasting.

Dr. Yan is an Editor of the *International Journal of Artificial Intelligence,* and an Editorial Board Member of the *International Journal of Prognostics and Health Management*.