An artificial neural network approach for short-term wind speed forecast

by

Pallab Kumar Datta

B.S., American International University-Bangladesh, 2012

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2018

Approved by:

Major Professor
Dr. Anil Pahwa

Copyright

Abstract

Electricity generation capacity from different renewable sources has been significantly growing worldwide in recent years, specially wind power. Fast dispatch of wind power provides flexibility for spinning reserve. However, wind is intermittent in nature. Thus, stable grid operations and energy management are becoming more challenging with the increasing penetration of wind in power systems. Efficient forecast methods can help the scenario. Many wind forecast models have been developed over the years. Highly effective models with the combination of numerical weather prediction and statistical models also exist at present. This study intends to develop a model to forecast hourly wind speed using an artificial neural network (ANN) approach for effective and fast operation with minimum data. The procedure is outlined in this work and the performance of the ANN model is compared with the persistence forecast model.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 Introduction

This document outlines a study of wind speed forecast model with an artificial neural network (ANN) approach known as nonlinear autoregressive neural network (NARNET). The model is developed for step ahead hourly wind speed forecast using historical data.

Wind is a free source of energy and wind power generation is environment friendly. The contribution of wind generated power in the grid has been significantly increasing worldwide in recent years. As of the end of 2016, the total installed wind power capacity worldwide amounted to nearly 486,790 MW with a growth rate of 12.5% from 2015 [1]. Figure 1.1 shows the growth of global cumulative installed wind capacity from 2001 to 2016 [2].

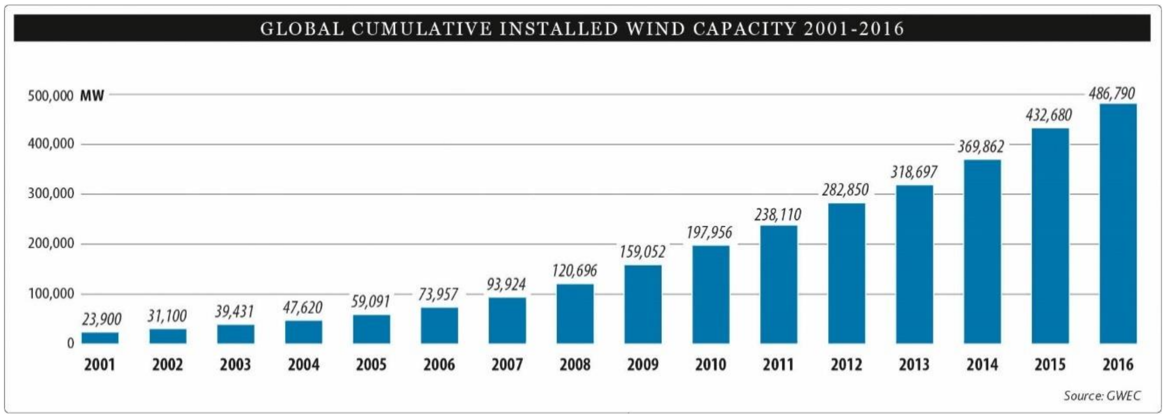

Figure 1.1: Global cumulative installed wind capacity from 2001 to 2016

Environmental benefits as well as incentive policies made wind power more and more popular in USA in recent years. In 2013, 4.13% of overall electricity generated in USA came from wind power, which would be sufficient to power 15.5 million American homes [3]. In 2016, the percentage of wind power (from utility scale facilities) in USA became 8% of the total capacity. However, due to low capacity factors of wind turbines, it contributed to approximately 5% of the overall generation in the same year; which is the highest for a renewable resource after hydropower [4]. The growth of installed wind capacity in USA from 2008 to 2015 is shown in Table 1.1 [5].

Table 1.1: Installed wind capacity in USA from 2008 to 2015

	Year							
	2008	2009	2010	2011	2012	2013	2014	2015
Total Installed Capacity (MW)	25, 410	34, 863	40,267	46,919	60,007	61, 108	65,754	74,347
Growth (%)		37.2	15.5	16.5	27.89	1.8	7.6	13

Figure 1.2 is a visual representation of the above growth.

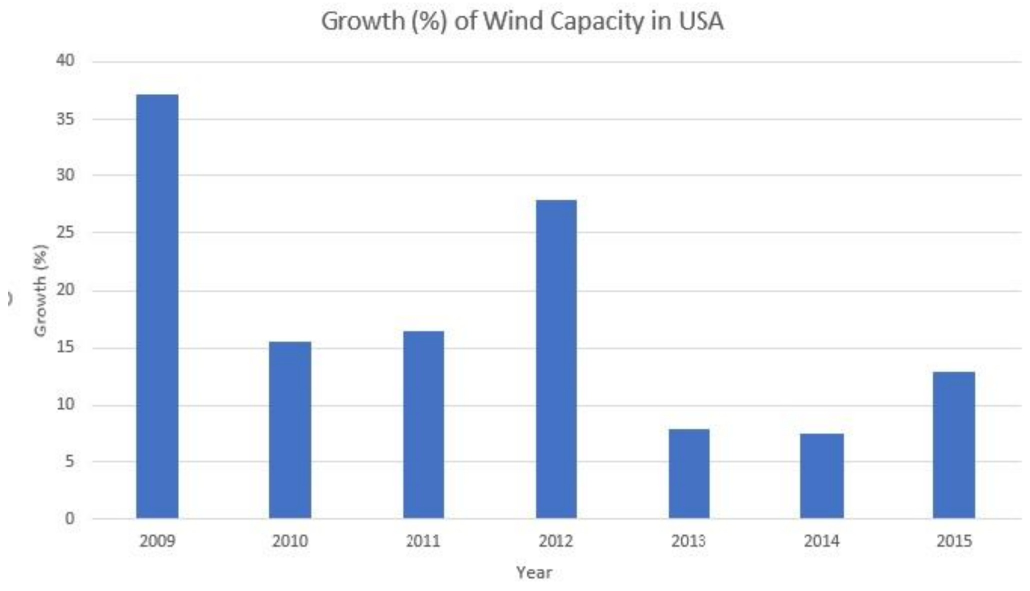

Figure 1.2: Growth (%) of wind capacity in USA from 2009 to 2015

Despite of installed capacity and advantages of clean and inexpensive production, desired generation from wind is not always readily available due to the intermittent nature of wind. Consequently, the power system operators are required to deal with largely fluctuating wind penetration which affects operation and reliability of the system. Efficient forecasting can significantly improve the situations involving stability of the system, dispatch and electricity market operations [6].

Several methods of wind speed and power forecast have been developed over the past years which can be classified in two main categories- statistical and physical methods [7]. Statistical models use historical data of wind farms to predict the future power generation outputs, while physical methods require different geographic, meteorological and technical considerations such as terrain structure, temperature, pressure, density and so on to determine wind speed, and give wind power prediction output from characteristics of the turbines [8]. Often the results of these models are combined with statistical models to improve the accuracy of the forecast, which are known as hybrid models [9].

Short term wind forecasting is a highly important area of research nowadays. The forecasting horizon can typically be few days to hours and minutes. Accurate prediction of wind behavior and wind generated power allows the system operators to deal with intermittent penetration of wind more reliably by providing better scheduling and dispatch. Long term forecasting is more significant to determine the trend and effects in energy markets [10].

This study incorporates a statistical method with an artificial neural network (ANN) approach to forecast short term wind speed. The introduction, background and objectives of the study are outlined in chapter 1. Chapter 2 describes the relationship between wind speed and power, and different methods of forecast. In Chapter 3 the ANN approach, namely the nonlinear autoregressive (NAR) model is discussed. Chapter 4 illustrates the methodology followed by results of the simulated experiments in Chapter 5. Chapter 6 discusses the possible improvements to the model and future work. Chapter 7 gives the conclusion of the project.

1.2 Background

By maximizing the utilization of the renewable resources, it is possible to reduce dependence on fossil fuels. Significant focus on biomass, geothermal, hydropower, solar and wind energies has been observed worldwide since last few decades. Wind power is becoming a very popular in recent years.

The power systems are required to handle higher penetration of wind power with its increasing contribution. Since wind flow is unpredictable in nature, decent forecasting methods can improve the quality of operation of wind farms, hence the power systems. It is possible to further reduce the cost of electricity and promote sustainable energy to higher extent by achieving these improvements.

Numerous wind forecast models have been developed over the last few decades. Many effective models are combinations of physical and statistical methods. However, developing and running those models frequently can be very expensive, specially when it comes to short-term or very short-term forecasting requirements to aid real time dispatching decisions. Effective statistical forecast methods can play a vital role in this scenario. Statistical models typically utilize historical data such as wind speed, solar radiation, electricity generation etc. Therefore, these models are faster to be trained and more convenient to run frequently. For individual research purpose, it was found quite difficult to obtain data from commercial wind farms. Hence publicly available meteorological data of wind speed from the National Oceanic Atmospheric Administration (NOAA) website are used in this study [11]. For the model development and implementation in this experiment, historical wind speed data were obtained from a weather station in Dodge City, Kansas.

1.3 Objective

The objective of this project is to develop a wind speed forecast model using an ANN architecture known as nonlinear autoregressive neural network (NARNET). The model uses univariate time series data (wind speed) to produce hourly wind speed forecast.

Chapter 2 - Wind Speed and Power Forecast

2.1 Relationship between Wind Speed and Power

The fundamental goal of a power system is to provide sufficient power to meet the demand at any given time. Load demands are always varying because of continuous switching on and off by the consumers, thus the objective becomes difficult to maintain. As wind is free source of clean energy, wind power is becoming more popular in recent years and integration of wind power in the grid worldwide has been growing every year. Although wind power has many advantages, the main disadvantage of wind is its uncertain nature. As the wind speed varies, so does the generation from a turbine accordingly. The theoretical relationship between wind speed and harvestable power at the turbine can be expressed by the following equation 2.1.

$$P = (1/2)\ C_p\ \rho\ A\ V^3 \qquad (2.1)$$

Where,

P = Harvested power

C_p = Capacity factor of the turbine

ρ = Air density

A = Area swept by the turbine blades

V = Wind speed

The above equation is satisfied only between the cut-in speed and rated-power speed of the wind turbine due to mechanical inertia and pressures on the turbine and generators. Wind power forecast can be generated based on the theoretical relationship and turbine specifications, if wind speed forecast is available. Another statistical approach to forecast wind power is based on utilizing power generation data for a similar forecast model.

2.2 Wind Power Forecast

Wind speed is a natural phenomenon and it is uncontrollable, hence the power produced by wind turbines is uncontrollable as well. Higher integration of wind power in the systems will cause more difficulties regarding the reliability, as the systems will be unable to control all generated power well. With high wind penetration (e.g. >5%), wind forecasts are especially essential for effective grid management [12]. The power systems have their reserve units to cover any variation in load demands, however given the uncertainty of contributions from the integrated wind turbines or farms, the spinning reserves will be required to satisfy different specification and as a result of that effective costs of the units are likely to be higher. These problems can further raise the price of energy following the regulations of energy market [13].

Power systems have been achieving goals of efficient and stable operation with load demand variabilities by load predictions. Similar approaches can be utilized for wind power generation, which will certainly ease the difficulties of large wind power integration into power systems. Proper forecasting can further advance the market and operation strategies of wind farms by improving the scheduling and management of generation units. Additionally, by managing wind power more efficiently it is possible to decrease the consumption of fossil fuel in the traditional plants. Subsequently electricity price is also possible to be reduced as the production cost, spinning reserve cost etc. would go lower.

2.3 Different Methods for Wind Forecasting

In last few decades several methods of wind power forecasting have been developed. The forecasting methods can be generally classified into two large categories – physical and statistical. Physical models, also known as numeric weather prediction (NWP) models, are primarily developed for large-scale area weather prediction [14]. These models consider terrain, obstacles, temperature, pressure etc. to predict wind speed at a future time. Although NWP models are well established through years of extensive research, when it comes to site specific and very short term and short-term forecasting, these models usually provide less accuracy [15]. Often NWP models utilize site specific numerical conversion equations and digital elevation models that produce more accurate forecast results.

Statistical methods provide forecast results by learning from past data or patterns. The past wind data may include wind speed, direction, temperature, power generation etc. as variables. Among the statistical methods the most basic approach is persistence model which are typically good for stable weather conditions. Persistence model states that the next value in time is same to the previous value. This can be expressed by equation 2.2 below.

$$P_{t+1} = P_t \tag{2.2}$$

Where,

P_{t+1} = next value in time

P_t = current observation (at time t)

Autoregressive (AR) models and autoregressive integrated moving average (ARIMA) models are found to outperform persistent models [14, 16]. With the emergence of artificial intelligence techniques, wind forecasting methods with artificial neural network, fuzzy logic system, support vector machines etc. have also been evolving in last few decades. It has been observed that ANN models performed reasonably in multistep-ahead prediction of mean wind speed [18]. Studies also showed that nonlinear autoregressive ANN models perform better compared to ARIMA models in multi-step ahead hourly wind speed prediction on several occasions [18].

2.4 Time Horizon of Wind Forecasting

The standard of time horizon for wind speed/power forecasting is quite equivocal. In many literatures, the forecast horizons were defined in different ways. The convention taken for this study is summarized below in Table 2.1 along with the applications of different forecast horizons [19].

Table 2.1: Forecast time horizons and applications

Type	Time Horizon	Application
Very short-term	5-60 minutes ahead	Operating reserve, real-time dispatch decisions
Short-term	1-6 hours ahead	Unit commitment for next hour operation, load following
Medium-term	1 day ahead	Day ahead unit commitment and scheduling, energy market trading
Long-term	Seasonal	Contingency analysis and resource planning

Chapter 3 - Artificial Neural Network Approach

3.1 Introduction

Artificial neural networks (ANN) are quite effective in solving nonlinear problems where inputs and outputs lack well-defined relationship. In ANN, related parameters are usually characterized by learning from sample data rather than following a fixed model. ANN are based on the concept of computation performed by human brain [20]. It can be termed as simplified imitation of biological nervous system consisting of highly interconnected units for parallel distributed processing. These units are called neurons. Weighted sum of inputs is produced in each neuron and a bias is added to it. Then the summation is passed through a thresholding unit or transfer function. Figure 3.1 shows the formation of a simple artificial neuron.

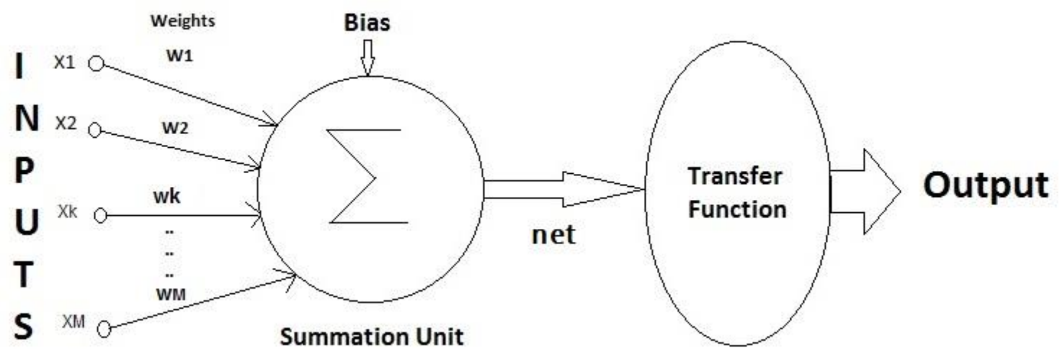

Figure 3.1: Simple model of an artificial neuron

The operation of a discrete neuron can be mathematically expressed with the following equations.

$$net = \mathrm{b} + \sum_{k=1}^{M} (\mathrm{wk.\,xk})$$

$$\mathrm{y} = \Theta(\mathrm{net}) = \begin{cases} 0, & net \leq 0 \\ 1, & net > 0 \end{cases} \tag{3.1}$$

Here, b = Bias

wk = weight between k-th input and the neuron xk = k-th input

ϴ = thresholding function

y = output of the neuron

σ = activation or transfer function

The threshold ϴ can be replaced with a differentiable nonlinear transfer function σ and equation (3.1.1) can the rewritten as y = σ (net). There are several types of transfer functions. Two commonly used nonlinear transfer functions are sigmoid and hyperbolic tangent functions.

Sigmoid function: Figure 3.2 shows sigmoid function. Sigmoid function can be expressed with the following equation.

$$\sigma(net) = \frac{1}{1+e^{-net}}, \qquad 0 < \sigma(net) < 1$$

Derivative: $\sigma'(net) = \sigma(net)(1- \sigma(net))$

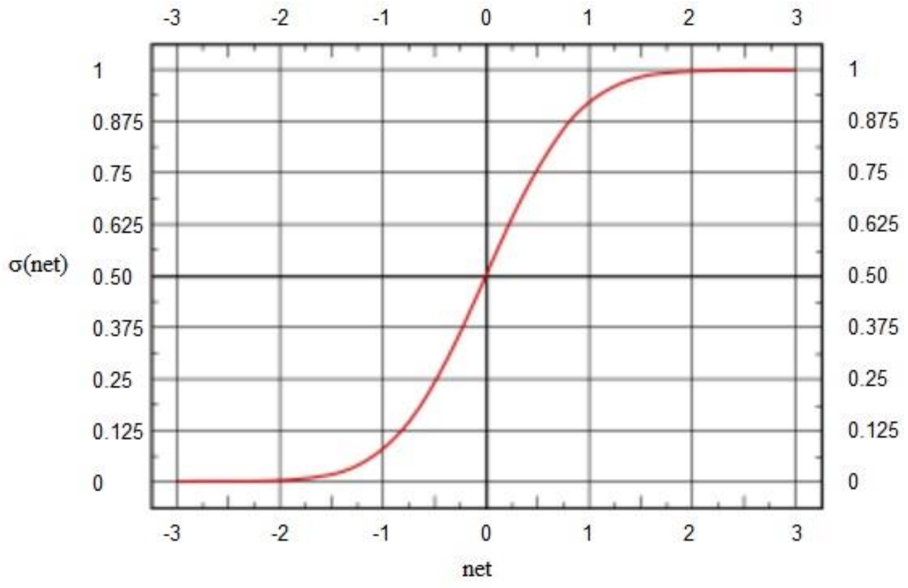

Figure 3.2: Sigmoid function

Hyperbolic tangent function: Figure 3.3 shows hyperbolic tangent function. The function can be expressed with the following equation.

$$\sigma(\text{net}) = \frac{1-e^{-net}}{1+e^{-net}}, \qquad -1 < \sigma(\text{net}) < 1$$

Derivative: $\sigma'(\text{net}) = 1 - \sigma^2(\text{net})$

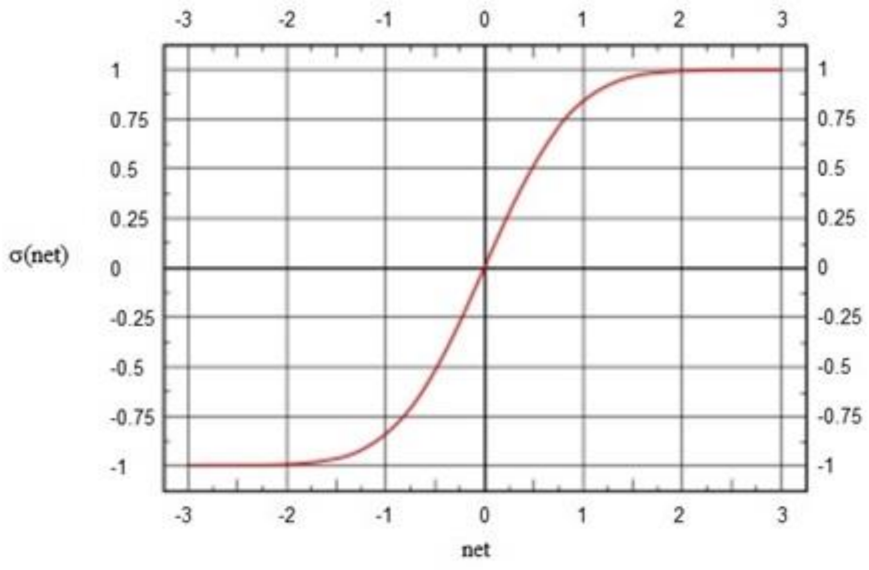

Figure 3.3: Hyperbolic tangent transfer function

3.2 Neural Network Architecture

There are three fundamental classes of neural networks: single layer feedforward network, multilayer feedforward network and recurrent network. Feedforward network is a formation where every input neuron is connected to output neurons through synaptic links carrying weights. The connections are not allowed in the opposite direction; hence they are called feedforward networks. Feedforward networks are of two types, namely single layer and multilayer networks. Single layer feedforward network consists of one input layer and one output layer. The computations are performed in the output layer only, therefore it is called a single layer network.

Multilayer feedforward network or multi-layer perceptron (MLP) structure have one or more hidden layers in between the input and the output layers. The hidden layers accommodate intermediate calculations in the units called hidden neurons before sending the inputs to the output layer. The weights assigned between input and hidden layers and between output and hidden layers are termed as input-hidden and hidden-output layer weights. Both flexibility and complexity of a network increases with increase in number of hidden neurons as well as number of layers [21]. The simplest form of MLP is a three-layer network. Sometimes it is termed as two-layered network since there is usually no computation performed in the input layer. In this document, the three-layer convention is used. This configuration has been found robust and specially suitable for forecasting purposes [22]. This structure efficiently allows the system to learn from retroactive data through supervised learning.

Figure 3.4 shows a generic model of three-layered feedforward ANN with L number of inputs, M number of hidden neurons and 1 output. W_{HX} and W_{YH} represent input-hidden weights and hidden-output weights respectively.

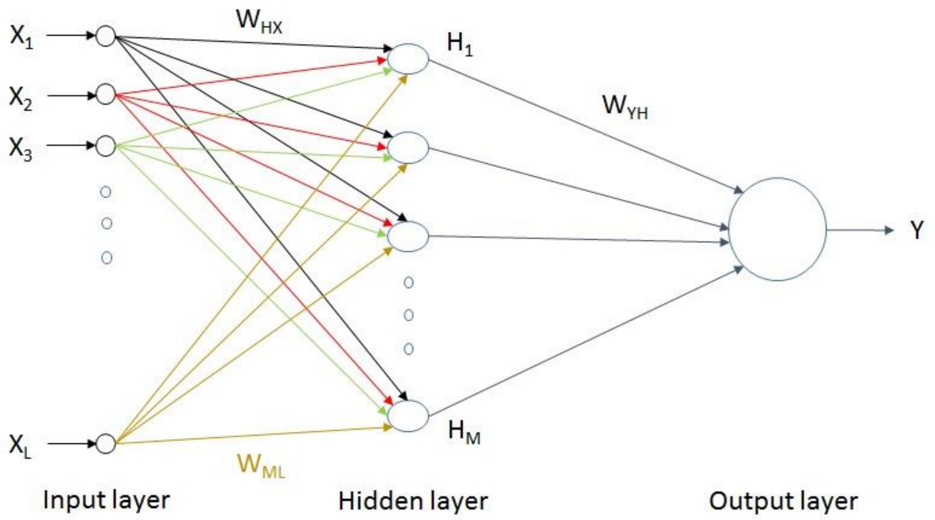

Figure 3.4: Generic model of three-layered feedforward ANN

In recurrent neural networks, there are feedback loops from the output layer to the input layer.

3.3 Learning Methods

There are two main types of learning methods in machine learning, which are supervised and unsupervised learning.

Supervised Learning: Supervised learning is a kind of machine learning algorithm which utilizes historical data, also known as training dataset, where each input data or pattern is associated with some output form. The algorithm forms a prediction model from this input-output relationship. From comparison of the network outcome and expected output, error of the model is determined. This error is then used to modify weights and biases to improve performance. Supervised learning is useful for classification and regression problems [23].

Unsupervised Learning: Unsupervised learning incorporates learning from dataset that do not have labeled responses associated with the input data. Unsupervised learning is usually used in cluster analysis, finding hidden patterns etc. [24].

3.4 Nonlinear Autoregressive Neural Network (NARNET)

NARNET is a type of dynamic neural network, suitable to for time series prediction using delays of a univariate time series. The architecture is a combination of multilayer perceptron and nonlinear filtering.

The prediction operation of NARNET can be mathematically expressed as a function of previous observed values. The expression can be written with the following equation.

$$y(t) = f\ \{\ y(t-1), y(t-2), \ldots\ldots , y(t-d)\ \} \tag{3.2}$$

Here,

$y(t)$ = the value in the series at time t,

d = number of delays

Open loop architecture is used to train NARNET. This architecture is similar to a three-layered feedforward structure described in section 3.2. If there are more associated variables in the model, the nonlinear autoregressive neural network with exogenous inputs (NARX) architecture can be used. Figure 3.5 shows a block diagram of NARNET, generated during MATLAB simulation.

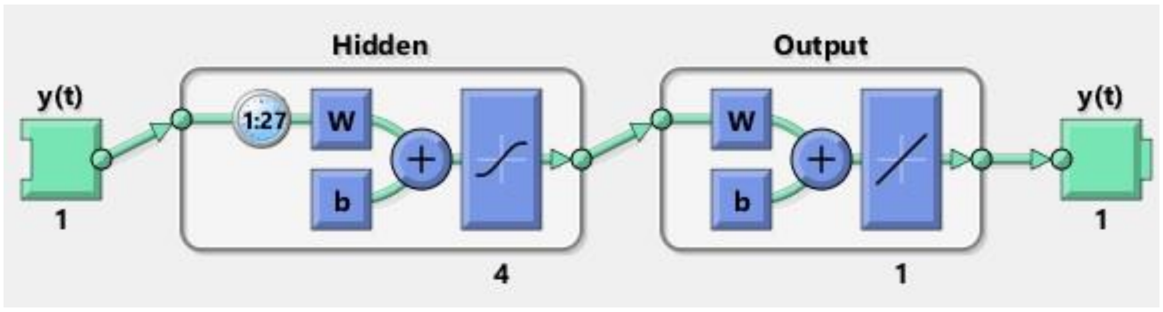

Figure 3.5: NARNET construction

In Figure 3.5, the block y(t) is the input series consisting of hourly wind speed observations. The number '1' at the bottom of the block indicates univariate time series. The series can be expressed as below. For simplicity of explanation, input series y(t) will be expressed as $y_i(t)$.

$$y_i(t) = y_{t-n}, y_{t-(n-1)}, y_{t-(n-2)}, \ldots\ldots\ldots , y_t \qquad (3.3)$$

Where,
y_t = observation at time t
n = number of observations

The hidden layer of the network is illustrated in the second block, namely 'Hidden'. The inner boxes 'w' and 'b' represent input-hidden weight and input-hidden bias respectively for a single neuron in the hidden layer. The term '1:27' denotes the number of delays used (27). The larger box after the summation sign indicates the transfer function of each neuron, as described in section 3.1. The number '4' at the bottom of the 'Hidden' block denotes the number of hidden neurons.

The 'Output' block in Figure 3.4 represents the output layer of the network. The inner boxes 'w' and 'b' represent the hidden-output weights and biases respectively. The transfer function of the output layer is linear. There is only one output neuron, which is denoted below the 'Output' block.

The last block y(t) represent the predicted output. This output y(t) is different from the input y(t) i.e. $y_i(t)$. Since the output of the network is a prediction of the input time series, MATLAB signifies both with the same variable. The output y(t) can be expressed with equation 3.3.

For example, if the input series contains observations of 100 hours and the delay is set to 10, the output of the network will be the predicted values for the last 90 hours of the input series. Based on these predicted outputs, the network can be used to forecast the value for the 101st hour and so on.

3.5 Network Parameter Selection

The selection of input size and tapped delay are described in chapter 4. For training of the NARNET, open loop feedforward structure is used, as seen in Figure 3.3. Hyperbolic tangent and pure linear functions are used as the transfer functions for the hidden and the output layers. Data division method is chosen to be in blocks of training, validation and test sets to maintain lag correlations.

There is no explicit explanation about how to choose the optimal number of hidden neurons in the hidden layers of a neural network. However, it is a common practice to keep this number as low as possible to ensure simplicity and robustness of the model. Simulations were run with different numbers of hidden neurons and best results were obtained with 4 hidden neurons for input size of 744 (hours).

3.6 Training Algorithm

A supervised method is generally used to train feedforward networks. A training set from historical data containing inputs and corresponding outputs are given to the network in this process. The success of training largely depends on the adequate selection of input for training. An ANN maps input and output relationship in the learning process by adjusting weights and biases to minimize error between produced output and desired output at each iteration. The iterations are repeated until the results converge.

Backpropagation algorithm is an efficient and most popular learning algorithm. In backpropagation algorithm, inputs are processed through the neurons to calculate final outputs and those results are compared with given outputs. The determined error is propagated back to the input and weights and biases in each layer are globally adjusted to minimize the error. Conjugate gradient algorithm is considered as a standard backpropagation algorithm incorporating sum of square error. However, it has been observed that Levenberg-Marquardt algorithm is capable to train an ANN much faster than gradient descent algorithm and considered as one of the most efficient training algorithms [25]. Therefore, Levenberg-Marquardt algorithm is used for the model developed in this project to train a three-layered feedforward ANN.

The Levenberg-Marquardt method can be mathematically expressed as following.

To minimize a function V(x) with respect to vector x, Newton's update is given by equation 3.4.

$$\Delta(x) = -[\nabla^2 V(x)]^{-1} \nabla V(x) \tag{3.4}$$

Where,

$V(x)$ = Sum of square error

$\nabla V(x)$ = Gradient vector

$\nabla^2 V(x)$ = Hessian matrix

The expressions can be given with the following equations.

$$V(x) = \sum e(x)^2 \tag{3.5}$$

$$\nabla V(x) = 2\, J^T(x)\, e(x) \tag{3.6}$$

$$\nabla^2 V(x) = 2\, J^T(x)\, J(x) + 2\, S(x) \tag{3.7}$$

The Jacobian matrix J(x) is given by equation (3.6.5).

$$J(x) = \begin{bmatrix} \frac{de1(x)}{dx1} & \cdots & \frac{de1(x)}{dxn} \\ \vdots & \ddots & \vdots \\ \frac{deN(x)}{dx1} & \cdots & \frac{deN(x)}{dxn} \end{bmatrix} \tag{3.8}$$

$$S(x) = \sum e(x)\, \nabla^2 e(x) \tag{3.9}$$

Neglecting the second order derivatives of the error vector, i.e. assuming S(x) = 0, the hessian matrix becomes:

$$\nabla^2 V(x) = 2\, J^T(x)\, J(x) \tag{3.10}$$

By substituting equation 3.6 and 3.10 into equation 3.4, the Gauss-Newton update is obtained as follows.

$$\Delta(x) = -[\, J^T(x)\, J(x)]^{-1}\, J^T(x)\, e(x) \tag{3.11}$$

The advantage of Gauss-Newton over the standard Newton's method is that it does not require calculation of second order derivatives. Another problem may arise that the Jacobian matrix and its transpose may not be invertible. Levenberg-Marquardt algorithm overcomes this issue by implementing the following update.

$$\Delta(x) = -[\, J^T(x)\, J(x) + \mu\, I]^{-1}\, J^T(x)\, e(x) \tag{3.12}$$

The learning rate parameter μ, is conveniently modified by the network during iterations of the algorithm. When μ is very small, the Levenberg-Marquardt algorithm acts as Gauss-Newton algorithm and provides faster convergence. When μ becomes higher, the 1st term inside the bracket in equation 3.12 becomes negligible with respect to the 2nd term inside the bracket and the algorithm acts as a steepest descent algorithm. Thus, the overall algorithm provides a balanced compromise between the speed of Gauss-Newton and convergence of steepest descent. For the simulations in this project, the learning rate is kept low (0.05) at the beginning.

The input data are divided into 3 sets – training, validation and test. Training set is used for learning and adjusting weights and biases. Validation set is used to prevent overfitting. It is crucial for a forecast model to avoid overfitting, otherwise it may fail to fit additional data or predict future observations reliably. When validation error starts to increase with iterations, the training process is stopped. These two sets are utilized to develop the model. However, the validation set errors do not have any impact on adjustments of weights and biases of the training set. Rather it is used only as criteria to stop training. The test set is excluded from model development. It is unseen by the network, hence used to determine performance of the network. The ratio used for training, validation and test sets is 70%, 15%. 15%. This data division is established to be efficient for neural networks to approach most of the problems.

3.7 Initialization of Weights and Biases

The weights and biases are initialized as small numbers between -0.1 and 0.1. The random number generations are controlled through ‘rng(n)’ command of MATLAB for 10 different initializations to perform 10 different simulations. Where, n = 1, 2, ……. , 10.

Chapter 4 - Methodology

4.1 Data Collection and Pre-processing

The wind time series data were collected from the NOAA website. The weather station selected is located in Dodge City, Kansas. The data contained hourly wind speed measurements of the location for 1 year, from January 2010 to December 2010.

Several assumptions were taken for the pre-processing of data. In some cases, there were multiple measurements for the same hour with an inconsistence interval of minutes. Those measurements were averaged to get a single value, excluding any instance containing extreme difference. To develop a forecast model for wind speed, it is important to deal with the extreme fluctuations like turbulence due to storm or any other natural phenomena. These measurements can unusually affect the generalization process, weights and biases of the neural network at the training stage which may lead to entirely wrong prediction. Extreme variations of such kinds which were not there for at least 2 hours were replaced by averaging the previous and the next mean measurements. This approach was applied if the differences between three consecutive observations were more than 12 m/s. From the nature of these kinds of fluctuations along with differences of wind speed at several previous and further hours, it was assumed that the observations can be affected by storms or any other natural phenomena. Also, there were several missing measurements which were filled by averaging as no detailed weather condition were available.

The selection of input size is important for the training of the neural network. Insufficient sample size can degrade the training process. On the other hand, excessive inputs can result into overfitting and misleading predictions. The number of sample data points (hours) for the model was chosen to be 744, which was obtained by trial and error with different sample sizes. Although input size may vary for location specific problems due to behavior of wind, several other works also implemented similar methods of adequate input selection for wind speed prediction models. Throwing raw data in the neural network resulted in poor training results, and the forecast outcomes were misleading. To improve the training process, the sample data were passed through a low pass filter to remove rapid shocks so that the network has improved ability for capturing the local trend.

To have a better idea about the changing pattern of the data set, statistical properties of the sample were analyzed in 3 durations – the entire sample (744 measurements), last 2 weeks (last 336 points) and last 2 days (48 points). A sample of the statistical properties showing actual and filtered series are shown in Table 4.1 and Table 4.2 below respectively, with a sample starting from January 1, 2010.

Table 4.1: Statistical properties of the actual sample

	Entire Sample	Last 2 Weeks	Last 2 days
Maximum	18.5	18.5	18.5
Minimum	0	0	0
Mean	5.0586	5.639	4.2437
SD	2.8997	3.0836	2.2629
Variation Coefficient	57.323	54.683	53.324

Table 4.2: Statistical properties of the filtered series

	Entire Sample	Last 2 Weeks	Last 2 days
Maximum	17.155	17.155	10.27
Minimum	-0.30488	0.26088	0.73324
Mean	5.3028	5.9112	4.4977
SD	2.9073	3.0747	2.2571
Variation Coefficient	54.826	52.015	50.184

The inconsistent oscillation of the sample series from the mean value within different intervals in Table 4.1 and Table 4.2 give some hint about the non-stationarity of the series. The term 'Variation Coefficient' used in the tables is a measure of relative variability of the series. It can be defined as following.

$$\text{Variation Coefficient} = (\text{Standard Deviation}) / (\text{Mean}) \qquad (4.1)$$

Filtering was performed as presenting raw data to the neural network resulted in highly erroneous training. Figure 4.1 shows the plots for the actual sample and filtered sample. It is difficult to visually examine the difference for the entire sample in a smaller space. Therefore, the first 100 points are shown in the figure below. The small difference between the red series (filtered) and the blue series (observed) is a visual indication that the statistical properties of the real data were not too much compromised due to filtering, which was explained from Table 4.1 and Table 4.2.

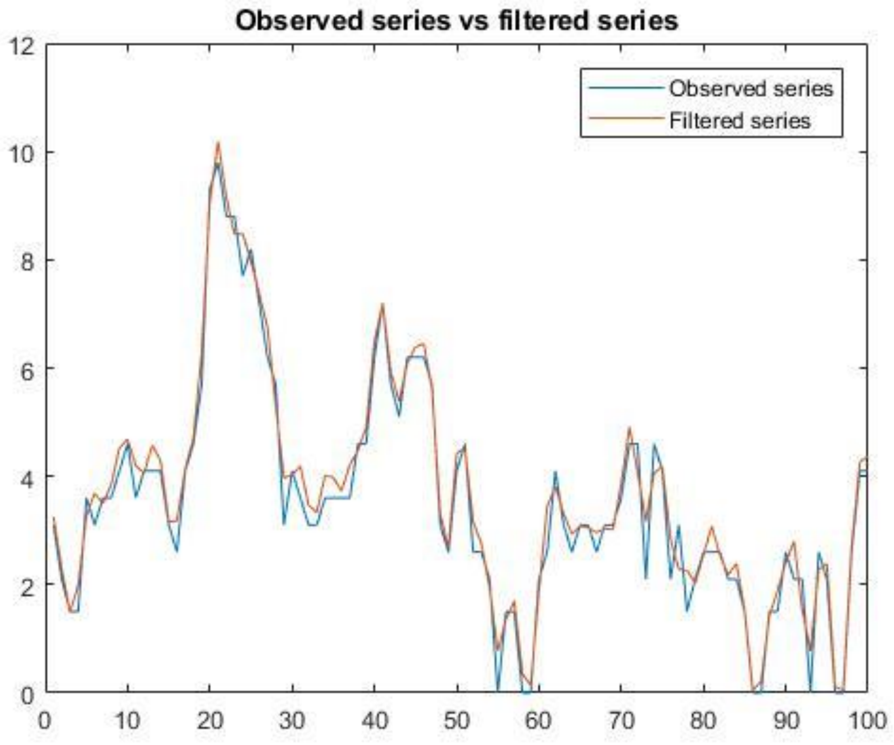

Figure 4.1: Observed series vs filtered series plot

4.2 Sample Autocorrelation Function

Sample autocorrelation function (ACF) of a series indicates the correlations of the series with its lagged values. For a series y = y_1, y_2, y_3, …. y_t, the sample lag-h autocorrelation can be given by the following equation.

$$ACF = \frac{\sum_{t=h+1}^{T}(\mathrm{y_t} - \bar{\mathrm{y}})\,(\mathrm{y_{t-h}} - \bar{\mathrm{y}})}{\sum_{t=1}^{T}(\mathrm{y_t} - \bar{\mathrm{y}})^2} \tag{4.1}$$

Here,

$\bar{y}$ = mean of the sample

y_t = value at time t

h = lag/delay

To determine the significance of a single lag -h autocorrelation, the error estimation can be given by equation 4.2.

$$\mathrm{E}_{\rho} = \sqrt{}[\,\frac{1 + (2\sum_{i=1}^{h-1} \rho_{\mathrm{i}}{}^2)}{\mathrm{N}}\,] \tag{4.2}$$

Here,

N = number of observations

Approximate 95% confidence intervals are at $\pm 2E\rho$. Figure 4.2 illustrates and example of autocorrelation function for the sample used in section 4.1.

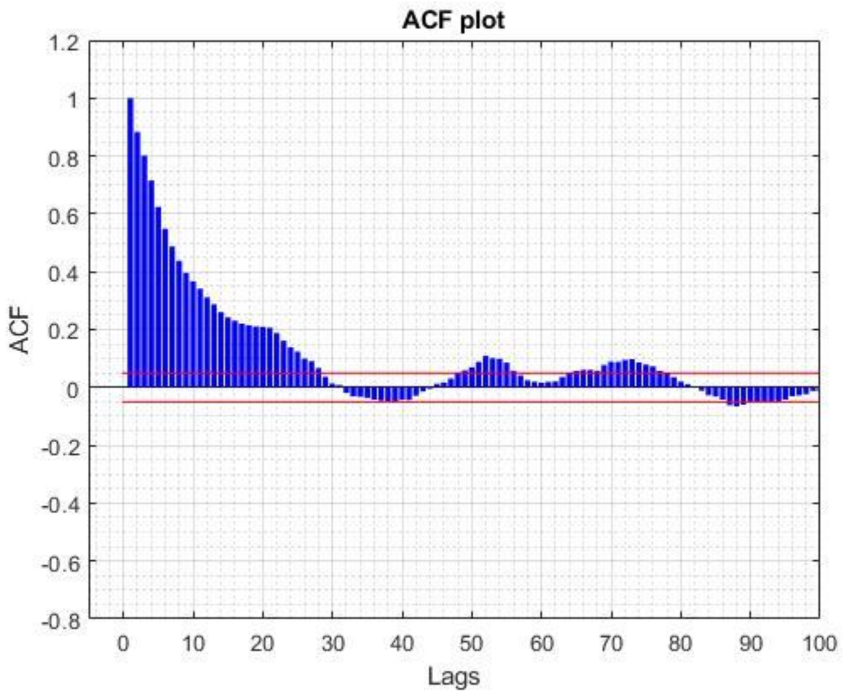

Figure 4.2: Sample autocorrelation function

The red horizontal lines in figure 4.2 indicate 95% confidence level. From the ACF plot, it is observed that the sample series is nearly non-stationary as strong correlations exist at comparatively high lags. Optimal lag for the NARNET is chosen from subsets of positive peaks higher than the confidence limit lines. For large datasets, complex error minimization algorithm needs to be formed in order to determine optimal lag, as there might be numerous options for better efficiency. Conventionally it is chosen from the subset of the 1st peak, which was followed in this study for simplicity. Seasonality is not observed in the sample series. For the stability of the forecast model, stationarity is important, specially when implementing ARIMA model. However, it was observed during the project that NARNET can deal with nonstationary time series.

In Figure 4.2, sample correlation falls below 95% confidence level at lag 29. Different lags before 29 were used to perform network training and the best result in terms of mean squared error (MSE) was observed at lag 28 for the above simulation. Lags after 28 did not improve MSE, rather the predictions went worse. Similar observations were seen during other simulations. All the simulations were performed in MATLAB 2017b.

Chapter 5 - Performance Analysis and Forecast Results

5.1 Performance Analysis

In the training algorithm, the mean squared error (MSE) is minimized and the test set MSE is used to evaluate performance of the NARNET model. The term can be expressed by equation 5.1.

$$\mathrm{MSE} = \left(\frac{1}{\mathrm{n}}\right) \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{yp_i} - \mathrm{yo_i})^2 \qquad (5.1)$$

Here, yo = observation at time t
yp = prediction for the same time period
n = number to observations

The performance of the NARNET model is compared with that of the persistence model. In the persistence model, the next predicted value is equal to the previous observation. This is a basic prediction model and found to perform very well in stable weather conditions. The statistical error measures evaluated for the comparison are MSE, mean absolute error (MAE) and mean absolute percentage error (MAPE). MSE is expressed in equation 5.1. The equations for MSE and MAPE are shown below.

$$\mathrm{MAE} = \left(\frac{1}{\mathrm{n}}\right) \sum_{\mathrm{i}=1}^{\mathrm{n}} |(\mathrm{yp_i} - \mathrm{yo_i})| \qquad (5.2)$$

$$\mathrm{MAPE} = \left(\frac{1}{\mathrm{n}}\right) \sum_{\mathrm{i}=1}^{\mathrm{n}} (|\mathrm{PE_t}|) \qquad (5.3)$$

Where,
$\mathrm{PE_t}$ = [(yp – yo)/yo]*100

MAE is a measure to identify the difference between a model and real observations as it measures the average of error’s absolute value. Although similar in nature to MAE, using MSE is more useful to handle optimization problems. MAPE can indicate higher degree of certainty to compare different models, specially when the other terms are relatively closer.

Several simulations were performed with different lag values. Figure 5.1 shows the ACF plot of the series and Table 5.1 shows the improvement of MSE with lags for one step ahead (hourly) forecast with data starting from January 2010. The forecasted value is the wind speed of the first hour of February 1, 2010 (hour 745 of the dataset).

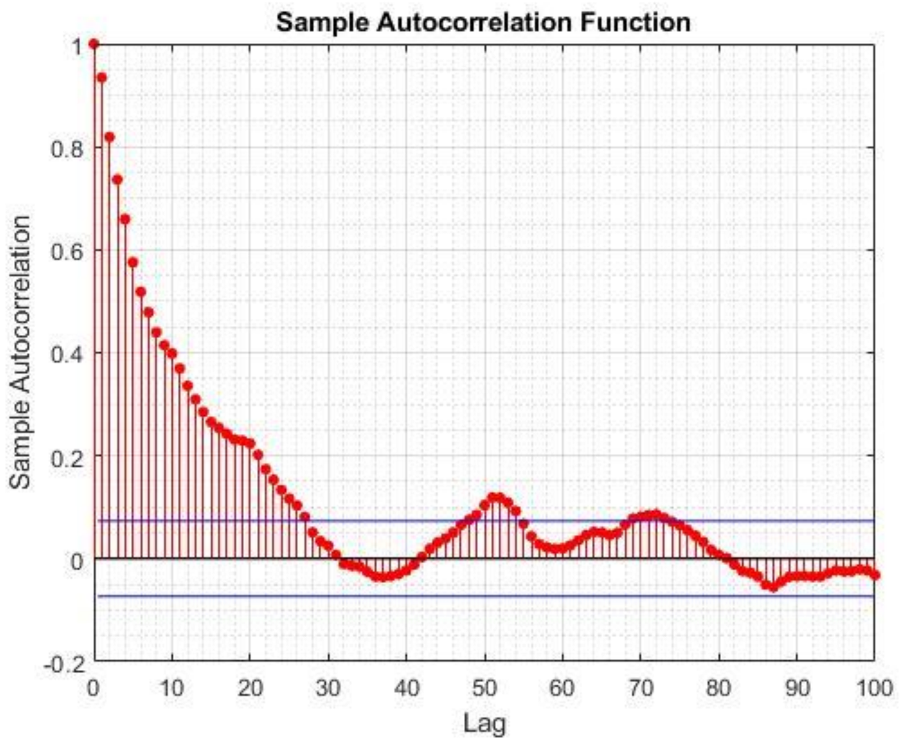

Figure 5.1: ACF plot of input data

The ACF plot shows correlations or dependence of the series with a delayed version of itself, as a function of delay. Lag in Figure 5.1 represent the sequence of observations from previous timesteps. For instance, sample autocorrelation at lag 10 indicates the significance of the (t-10)th observation to obtain the observation at time t. It is learnt during the project that the optimal delay for the nonlinear autoregressive neural network should exist among the subset of lags containing sample autocorrelation peak higher than 0.20. MATLAB code can also generated to find the subsets of these desired lags. Table 5.1 shows the improvement of errors in training, validation and test sets with the increase of lags.

Table 5.1: Simulation performance

	Lag	MSE (train)	MSE (validation)	MSE(test)
Starting point: 1 Input size: 744	1	0.0030	0.0082	0.0035
	2	0.0050	0.0107	0.0047
	3	0.0018	0.0045	0.0023
	4	0.0012	0.0035	0.0015
	5	9.08×10^{-4}	0.0024	0.0012
	6	6.66×10^{-4}	0.0020	9.12×10^{-4}
	7	5.01×10^{-4}	0.0015	6.41×10^{-4}
	8	4.11×10^{-4}	0.0014	5.34×10^{-4}
	9	3.29×10^{-4}	0.0011	4.48×10^{-4}
	10	2.63×10^{-4}	0.0013	4.11×10^{-4}
	11	2.08×10^{-4}	6.90×10^{-4}	3.17×10^{-4}
	12	1.83×10^{-4}	6.73×10^{-4}	2.72×10^{-4}
	13	1.33×10^{-4}	5.91×10^{-4}	2.08×10^{-4}
	14	1.04×10^{-4}	5.81×10^{-4}	1.76×10^{-4}
	15	1.27×10^{-4}	3.74×10^{-4}	1.85×10^{-4}
	17	1.62×10^{-4}	7.29×10^{-4}	2.27×10^{-4}
	18	1.36×10^{-4}	4.48×10^{-4}	1.97×10^{-4}
	19	1.17×10^{-4}	3.83×10^{-4}	1.75×10^{-4}
	20	8.68×10^{-5}	4.25×10^{-4}	1.51×10^{-4}
	21	3.23×10^{-5}	1.67×10^{-4}	7.04×10^{-5}
	22	6.79×10^{-5}	2.18×10^{-4}	8.81×10^{-4}
	23	2.08×10^{-4}	4.45×10^{-4}	2.18×10^{-4}

From Table 5.1 it is observed that the best performance of the model is obtained at lag 21 in terms of training, validation and test MSE values.

5.2 Simulation Results

Simulations were performed with 10 randomly initialized weights and biases for the same input data. These weights and biases were generated between -0.1 and 0.1. The random generations were controlled through rng() command in MATLAB to reproduce the results. For example, rng(1) generated a set of values within the above range. Similarly rng(2) generated a different set of numbers. The function rng can be also set to ‘default’. The network worked properly without setting the random number generator, but it was done to reproduce the results for comparison. It was observed that different initialization of input node weights and biases impacted the outputs. Simulation results are illustrated in this section.

Simulation 1:

Table 5.2: Simulation 1 summary

Sample starting point (hour)	Sample size	Forecast point (hour)	Significant lag	MSE (test set)	Observed Wind Speed (m/s)	Forecasted Speed (m/s)	Step-ahead forecast error (%)
7	744	751	20	1.47×10^{-4}	8.6	8.03	6.93

Figure 5.2 shows the plot of predicted outputs vs observed values. For visual clarity, prediction results for the last 40 hours are shown in the figure.

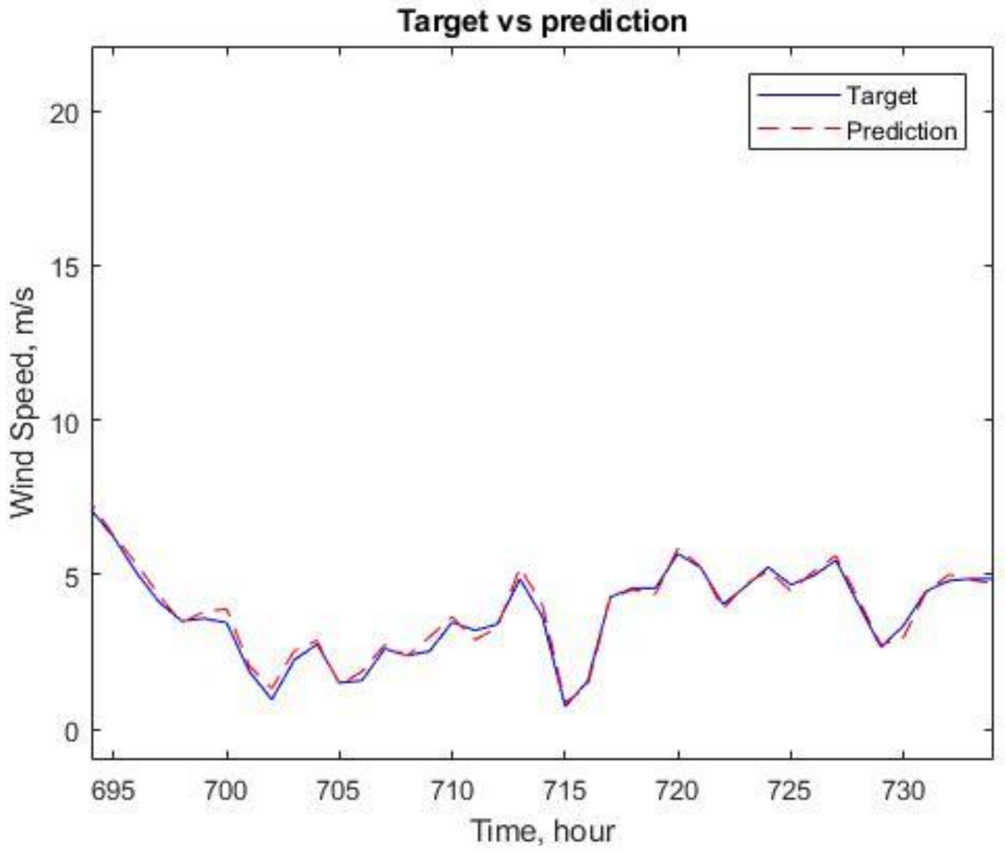

Figure 5.2: Output prediction (simulation 1)

Figure 5.3 shows the epoch number to obtain the best validation result (circled), followed by next epochs where the validation error failed to improve. This criterion determines when to stop the training process to avoid overfitting. In this case training is stopped after iteration 55. The best result in was obtained at epoch 49.

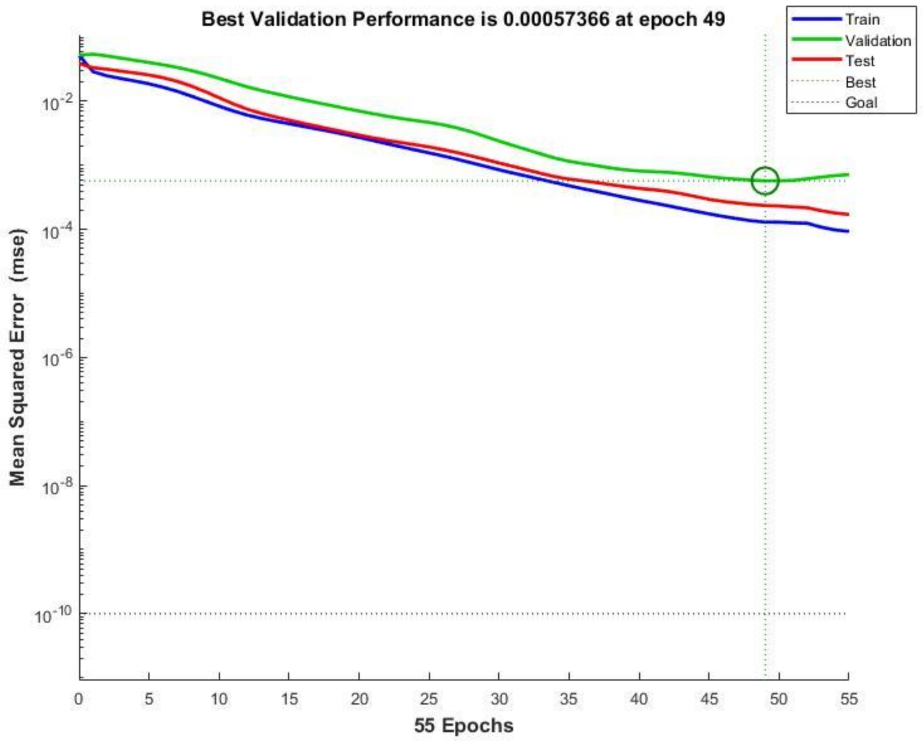

Figure 5.3: Best Validation Performance

Figure 5.4 shows the regression plot of the predicted series. A perfect regression should have the value R = 1. Figure 5.4 was obtained for the nonlinear regression where the outputs, i.e. the prediction at time t (for current observation), was responsive to previous 20 observations as the feedback delay of the network was defined as 20 for this case.

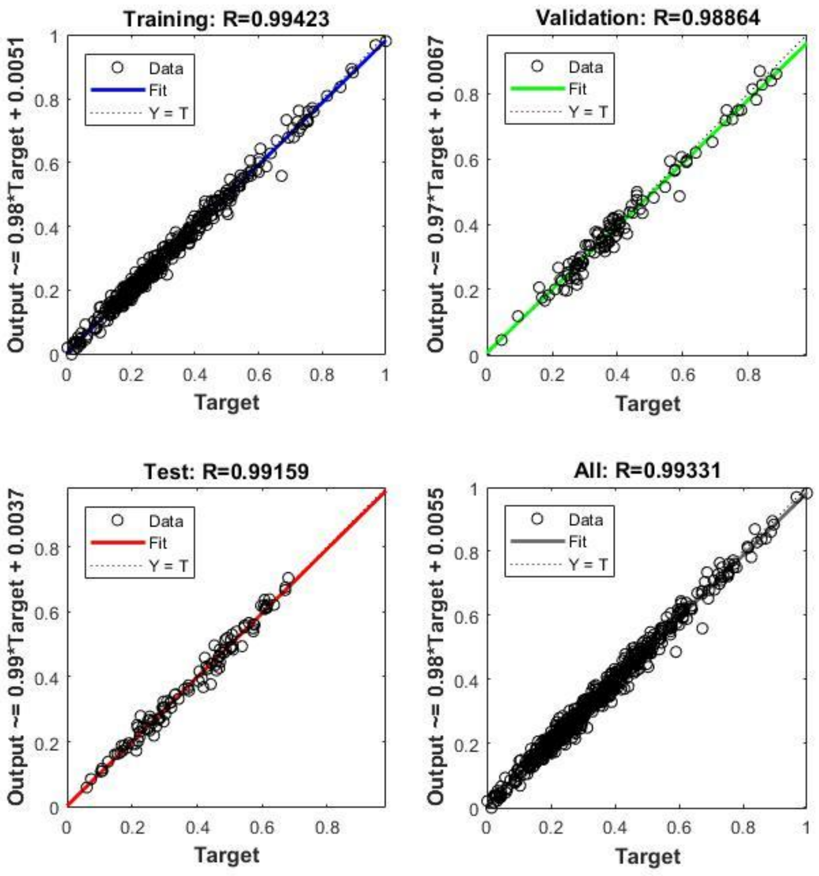

Figure 5.4: Regression plot

Figure 5.5 shows the error histogram plot. This figure shows the distribution of errors in the training, validation and test stages. For more efficient forecast model, it is desired that these errors will follow a normal distribution. This was not perfectly obtained, which indicates requirement of improvement for the model.

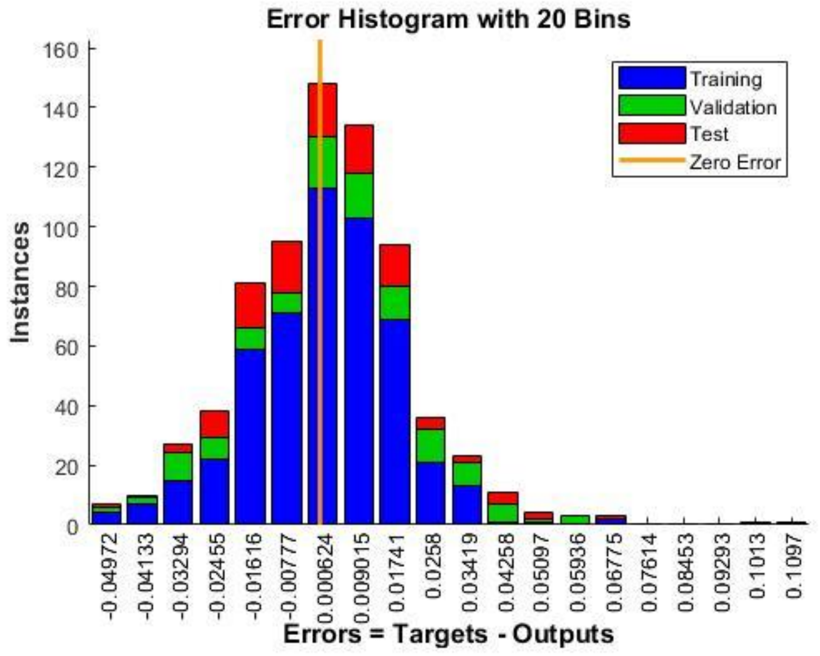

Figure 5.5: Error histogram plot

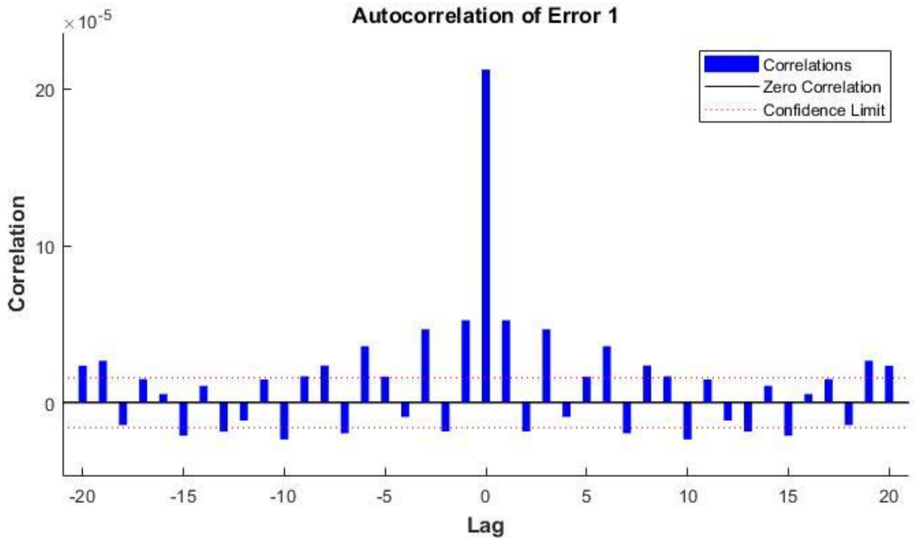

Figure 5.6: Error correlation plot

Figure 5.6 shows the error co-relation plot. It indicates significant corelations of the error with higher lag. These errors are prediction errors at the epoch during which training was stopped by the validation criterion. The lags represent the time steps of the observation for corresponding predictions. For a good forecast model, the errors should be uncorelated in time. This indicates that improvements are required for the reliability of the model. Some scopes of improvements are suggested in chapter 6. The graph also indicates stronger impact of correlations, which occured during filtering. Prediction pattern can be acheieved with lower significnt lag when the series is not filtered, but that results in higher value of traininng error. This problem can probably be solved by introducing lower weights and biases or modifying the error function to have smoother response of the network. The purpose of this study is to develop the basic formation of the forecast model. Implementation of the suggested improvements rquire further in depth analysis, hence included in future work possibilities.

Figure 5.7 shows the forcast vs observed wind speed graph for forecast horizon of 8 hours.

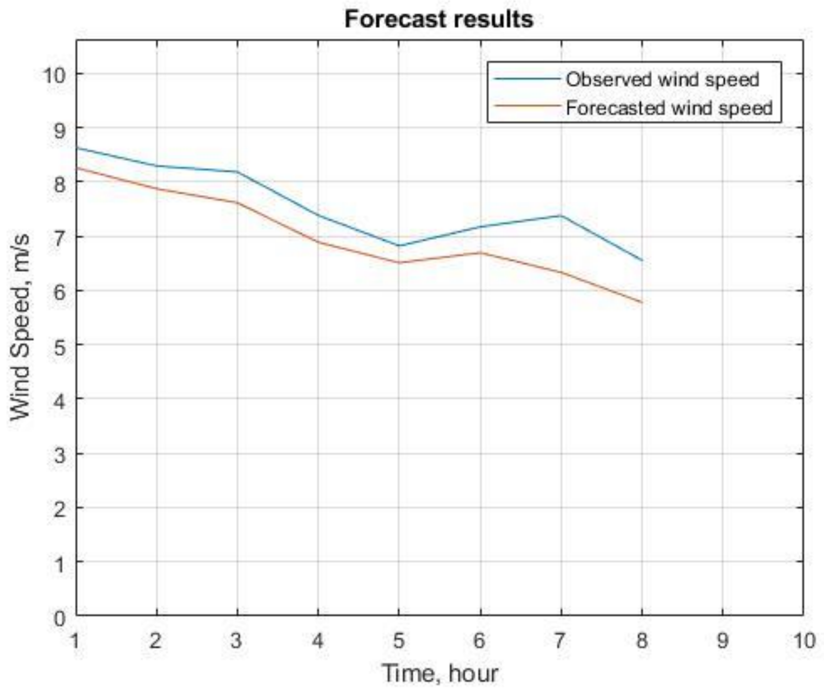

Figure 5.7: Forecast result for 8 hours horizon

Figure 5.8 shows the point errors for forcast horizon of 8 hours.

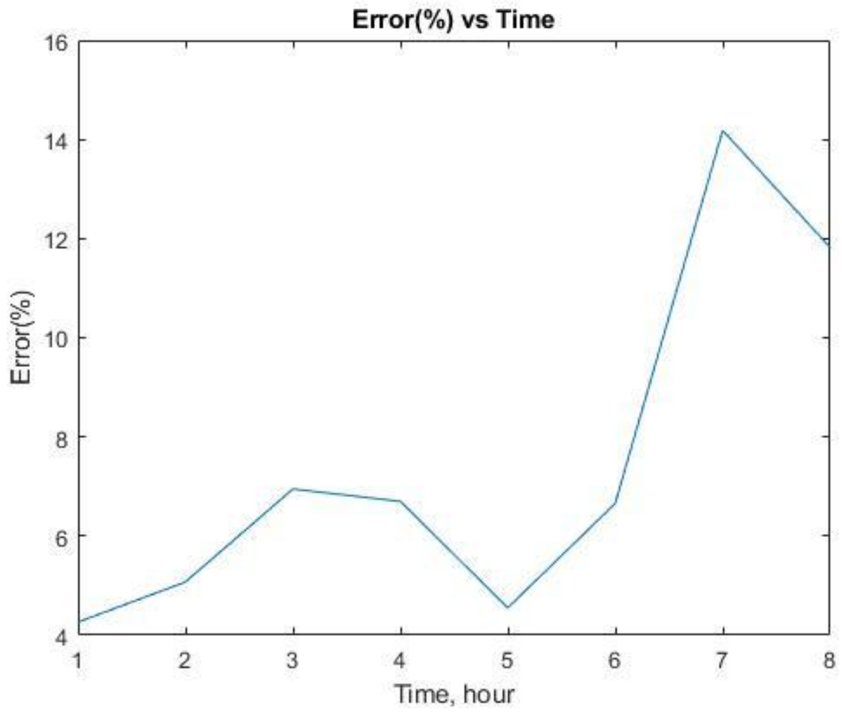

Figure 5.8: Error for forecast horizon of 8 hours

For the same data set used in simulation 1, further simulations are performed with different initializations of small weights and biases, generated by the random number generator in MATLAB. The simulation results are illustrated below.

Simulation 2:

Table 5.3: Simulation 2 summary

Hour	MSE (test set)	Observed Wind Speed (m/s)	Forecasted Wind Speed (m/s)	Step-ahead forecast error (%)
751	1.89×10^{-4}	8.6	8.2	4.65

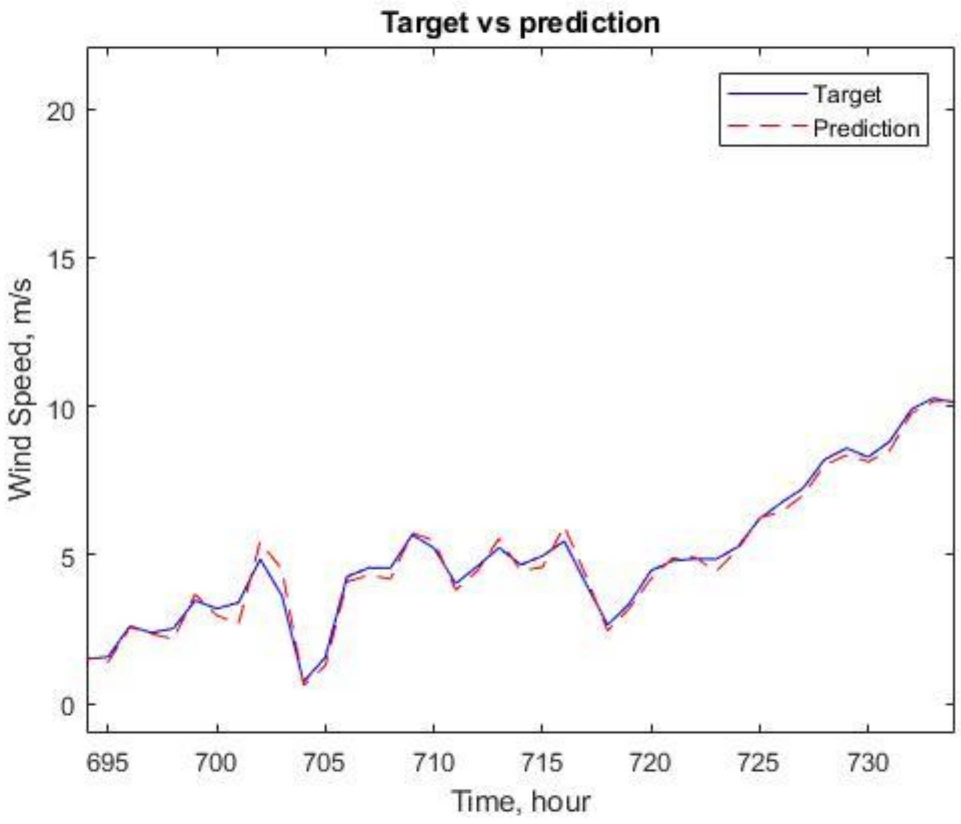

Figure 5.9: Output prediction

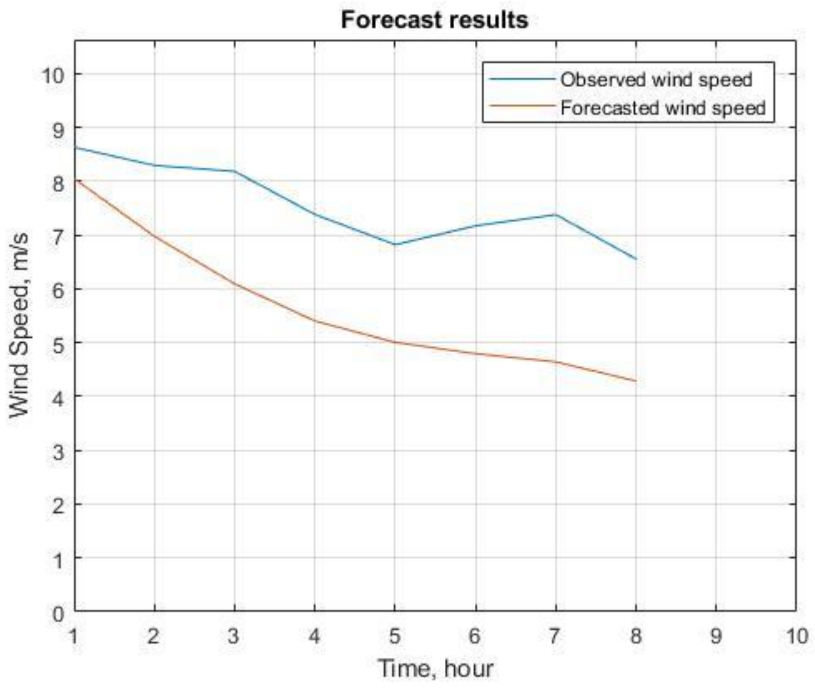

Figure 5.10: Forecast result for 8 hours horizon

Figure 5.11 shows the point errors for forcast horizon of 8 hours.

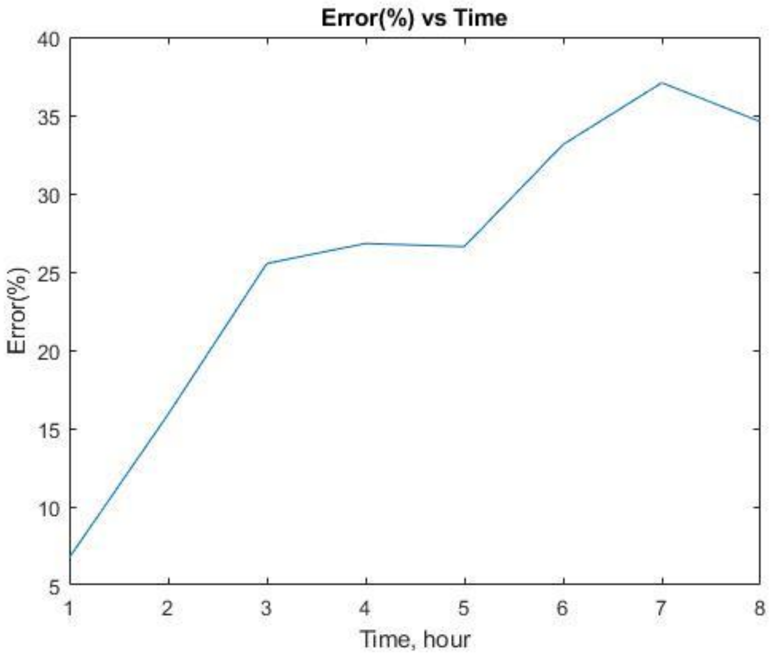

Figure 5.11: Error for forecast horizon of 8 hours

Simulation 3:

Table 5.4: Simulation 3 summary

Hour	MSE $\times 10^{-4}$ (test set)	Observed Wind Speed (m/s)	Forecasted Wind Speed (m/s)	Step-ahead forecast error (%)
751	8.61	8.6	7.9	8.14

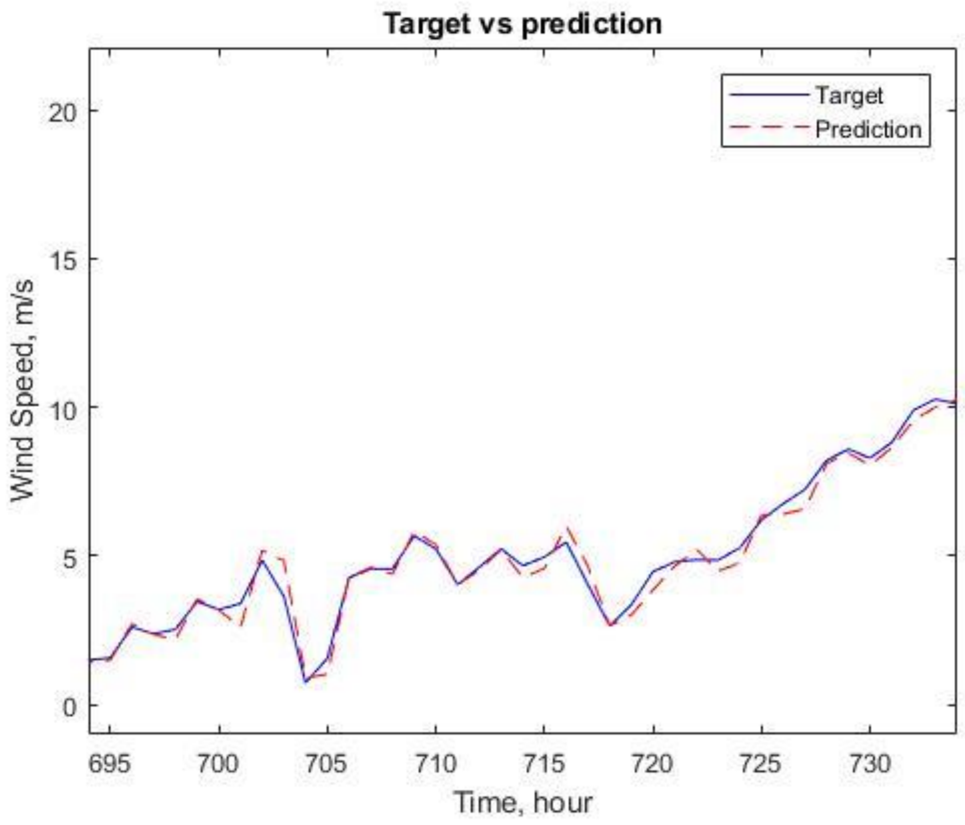

Figure 5.12: Output prediction

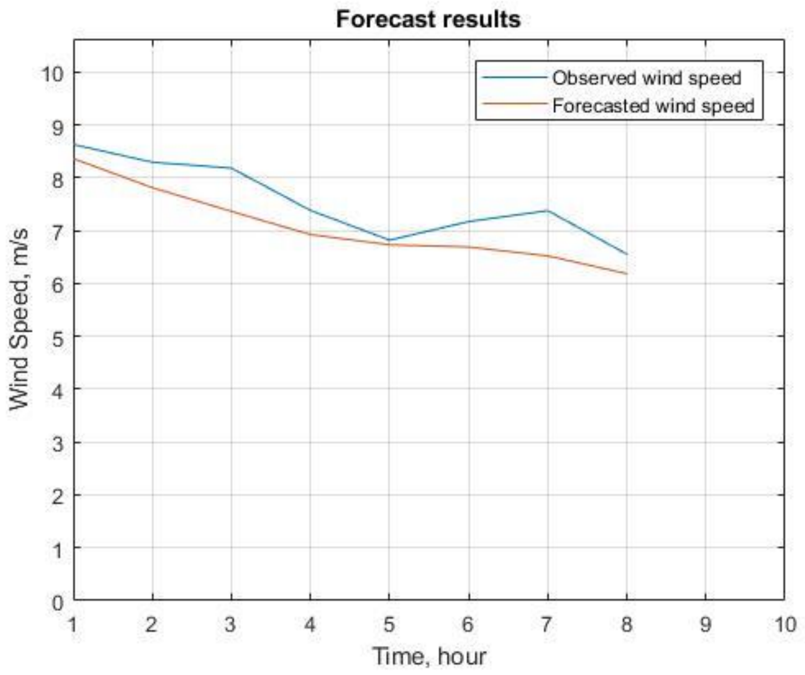

Figure 5.13: Forecast result for 8 hours horizon

Figure 5.14 shows the point errors for forcast horizon of 8 hours.

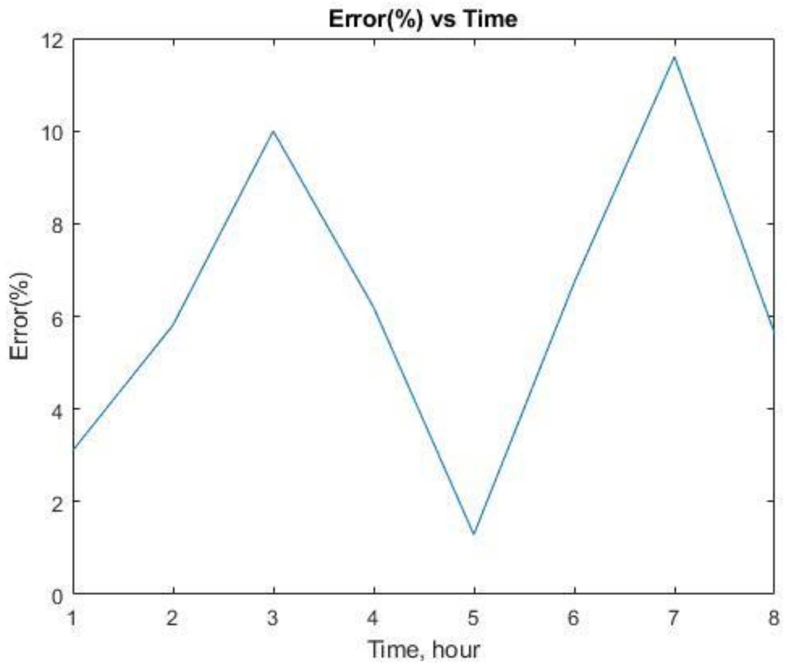

Figure 5.14: Error for forecast horizon of 8 hours

Simulation 4:

Simulation 4 was performed with the same input data and the same forecast points as the first three simulations, but the initialization of weights and biases was done with ‘rng(4)’, as explained in section 5.2. With this different initialization, the output prediction (Figure 5.15) and the forecast result (Figure 5.16) become slightly different from other simulations. All the simulations were performed in this manner.

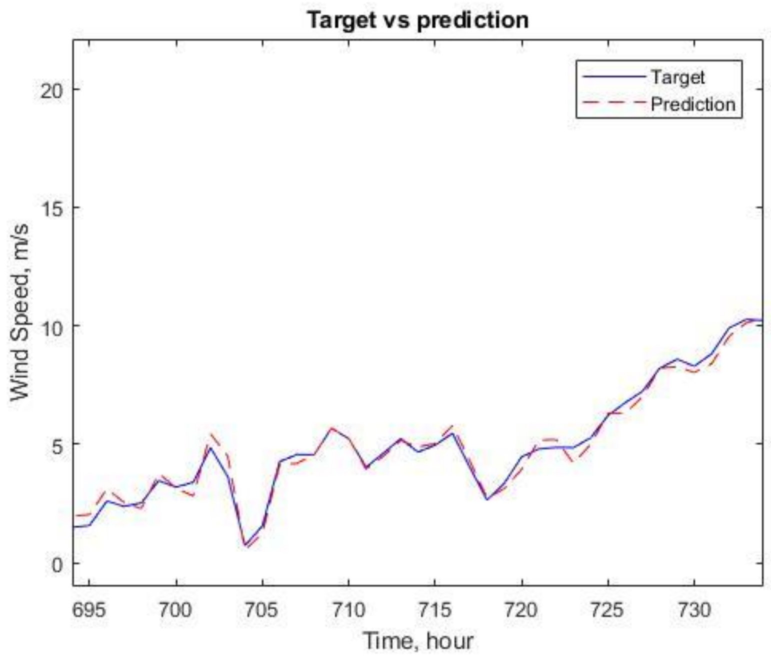

Figure 5.15: Output prediction

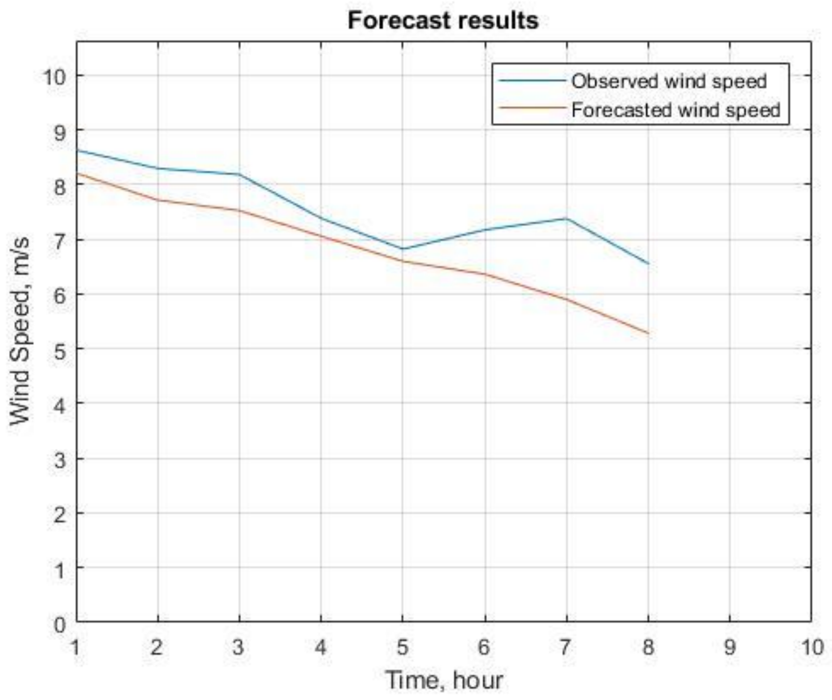

Figure 5.16: Forecast result for 8 hours horizon

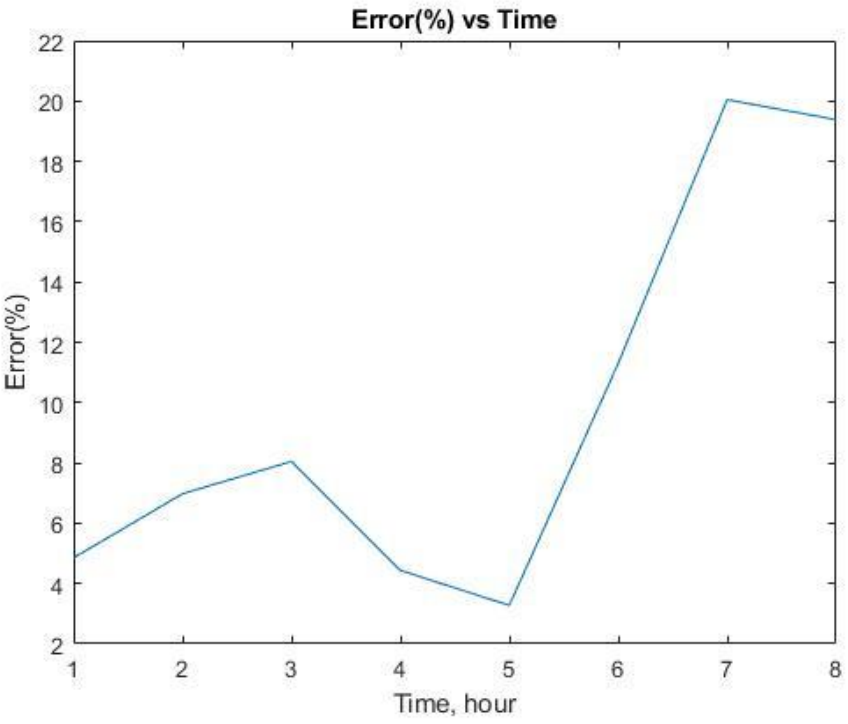

Figure 5.17: Error for multi-step ahead point forecast

Simulation 5:

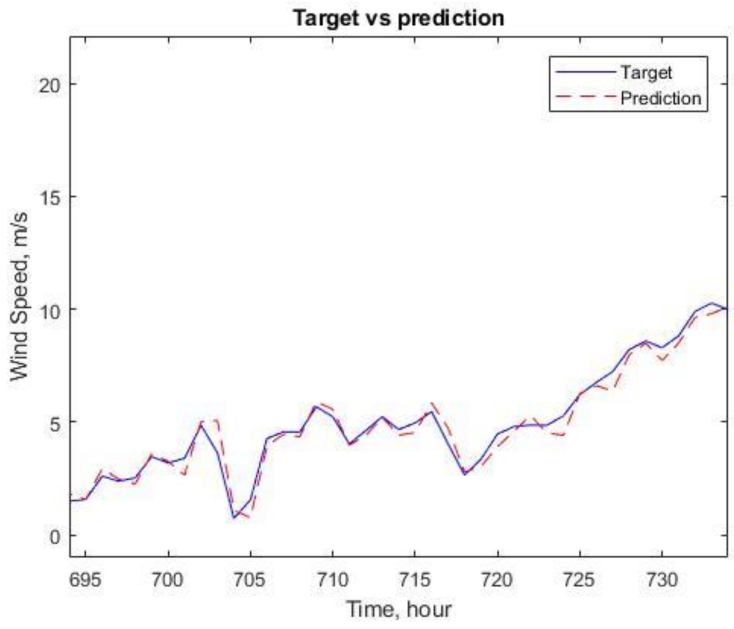

Figure 5.18: Output prediction

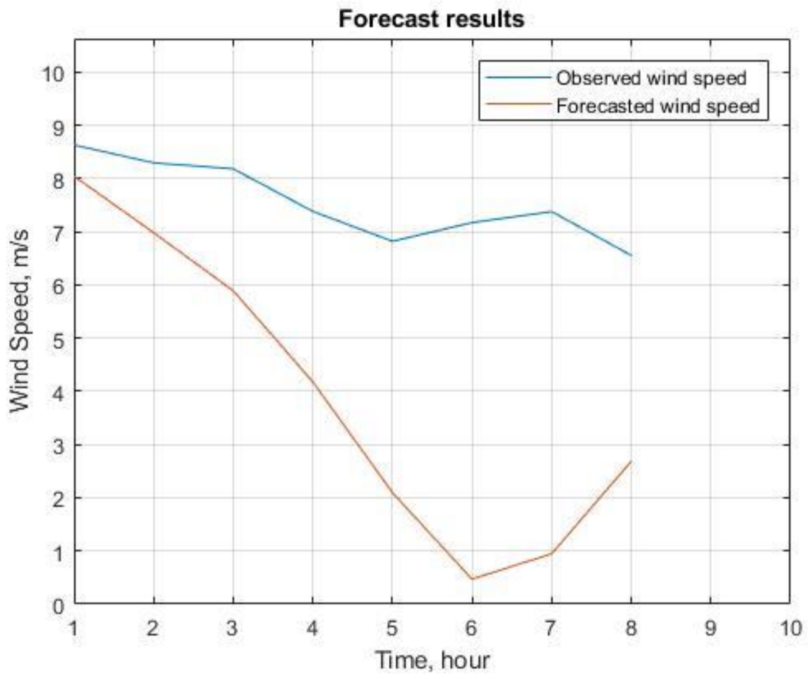

Figure 5.19: Forecast result for 8 hours horizon

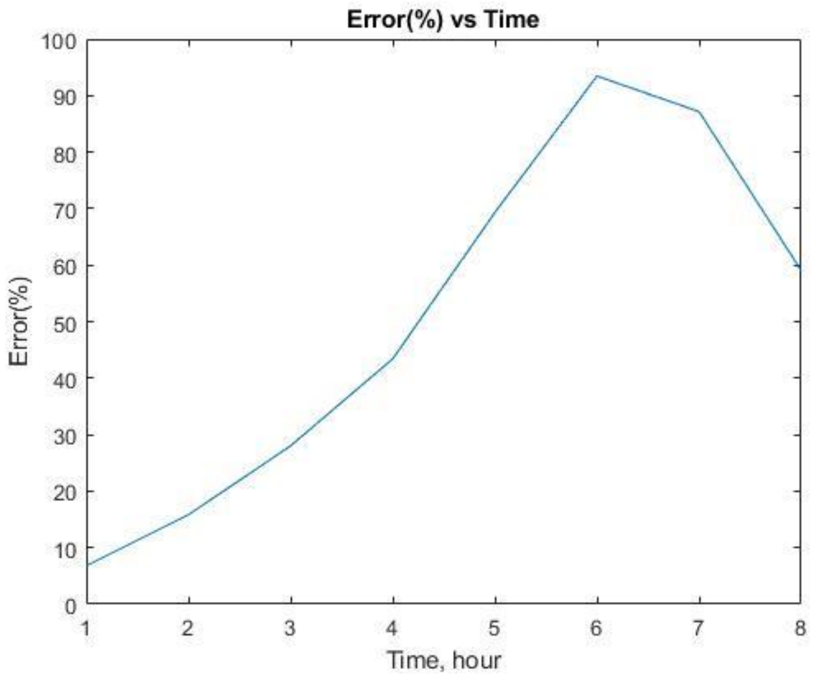

Figure 5.20: Error for multi-step ahead point forecast

Five additional simulations were performed with random initializations of the weights and biases for the neural network which gave similar results. From the simulations results above, it can be concluded that the implemented model is quite decent for 1-2 hours ahead wind speed forecast. However, the error correlations and the MSE values indicate requirements of improvement in the model. Forecast for further hours contained large errors, hence the model cannot be considered reliable for longer forecast horizon.

5.3 Statistical Analysis

Table 5.5 summarizes the percentage of point forecast errors and corresponding MSE of the test sets, obtained from simulations described in the previous section. Table 5.6 shows the standard deviations of the errors.

Table 5.5: Error percentage for forecast horizon of 8 hours

	`Run Number										
Hour	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	Mean
1	4.26	6.74	3.10	4.85	6.79	5.99	6.44	4.54	7.78	2.64	5.30
2	5.07	15.89	5.80	6.97	15.82	12.92	14.56	5.75	21.73	2.04	10.67
3	6.94	25.53	9.99	8.04	28.01	21.81	23.67	7.25	38.83	7.99	17.80
4	6.69	26.81	6.19	4.43	43.40	26.67	27.67	8.26	42.25	18.06	21.04
5	4.55	26.61	1.28	3.27	69.28	31.72	35.46	12.94	29.29	31.52	24.59
6	6.65	33.13	6.72	11.30	93.45	42.20	52.09	20.35	20.68	37.63	32.40
7	14.17	37.07	11.64	20.04	87.15	54.12	65.54	27.36	19.19	39.11	37.53
8	11.83	34.58	5.62	19.38	59.02	61.98	70.09	28.90	5.19	60.25	35.68
MSE (test)	1.68 $x10^{-4}$	5.65 $x10^{-5}$	6.14 $x10^{-4}$	1.33 $x10^{-4}$	5.99 $x10^{-5}$	1.08 $x10^{-4}$	1.08 $x10^{-4}$	1.62 $x10^{-4}$	2.15 $x10^{-4}$	2.08 $x10^{-4}$	

Table 5.6: Standard deviations of errors

Hour	Standard Deviation of Errors
1	1.70
2	6.36
3	11.24
4	14.65
5	20.48
6	26.43
7	24.68
8	25.25

From Tables 5.5 and 5.6 it can be inferred that 1 hour ahead wind speed forecast result using the developed model is quite decent. The performance degrades as the forecast horizon is extended. The lowest MSE value in the test set is obtained for run 2. Therefore, Range for the 1st deviation = mean ± standard deviation = 3.60 to 7.00. The obtained error for the 1st hour forecast for run 2 is 6.74, which falls within the range of the 1st deviation. Table 5.7 shows the 1st hour errors, sorted in ascending order of test set MSE.

Table 5.7: Forecast errors in ascending order of MSE

Sl	Run	MSE	Forecast error(%)
1	2	5.65×10^{-5}	6.74
2	5	5.99×10^{-5}	6.79
3	7	1.08×10^{-4}	6.44
4	6	1.08×10^{-4}	5.99
5	4	1.33×10^{-4}	4.85
6	1	1.68×10^{-4}	4.26
7	8	1.62×10^{-4}	4.54
8	10	2.08×10^{-4}	2.64
9	9	2.15×10^{-4}	7.78
10	3	6.14×10^{-4}	3.10

From Table 5.7, the first seven errors fall within the 1st deviation while rest of the values fall within the 2nd deviation. This indicates that the error tendency is likely to follow normal distribution pattern.

5.4 Comparison between NAR and Persistence Models

Comparison between the NAR model and the persistence model in terms of error standards described in Section 5.1, are summarized in Table 5.8 below.

Table 5.8: Comparison between NAR and Persistence models

Model	**MSE**	**MAE**	**MAPE**
NAR	1.28×10^{-4}	0.0083	3.4962
Persistence	0.0026	0.0404	15.0156

The statistical measures for the NAR model are obtained from simulation 7 of the previous section. From the statistical parameters in Table 5.8, it can be concluded that the developed NAR model performed better than the persistence model to forecast hour ahead wind speed at Dodge City.

Chapter 6 - Conclusion

6.1 Conclusion

With the emergence of renewable energies at present, efficient forecast methods are becoming more and more crucial to deal with intermittent natures of natural resources. Proper forecasting methods are no less important in the other energy and renewables related areas such as price forecast, solar radiation forecast, economic evaluations etc. The ANN approaches are convenient for efficient and frequent implementations. The nonlinear autoregressive model can be effective to forecast wind power, solar radiation and similar other problems.

A model is developed in this study using artificial neural network primarily to forecast step-ahead wind speed. The main problems faced during this project was collecting quality data and lack of sufficient documentations of the methods regarding the implemented architecture. Commercial wind firm data were not available in any public domain. Therefore, meteorological data from the National Oceanic and Atmospheric Administration (NOAA) website were used in this project.

The forecast model is developed with NARNET, utilizing univariate time series (hourly wind speed). The model is intended to work with minimum availability of statistical data to provide effective, fast and frequent implementation. These types of models can be trained and run much faster than the physical models. These are also cost effective as the approach is statistical and requires measurements of fewer variables. Similar approach to can be applied to other problems involving time series analysis, since the method is data driven.

Performance of the NARNET model is evaluated in terms of mean squared error (MSE), and compared with a persistence model. In the comparison it is observed that the ANN approach outperformed the persistence model to forecast hourly wind speed. In the simulations to forecast wind speed of any random hour, the developed model showed decent response. However, there are several aspects of the model, subject to further improvement, as discussed in section 6.2.

6.2 Observations and Possible Improvements of the Model

There are several scopes to improve the forecast model. Some of the possibilities are discussed in this section.

A general observation during this study was importance of the quality of data to develop an efficient model. Faulty measurements of inputs are likely to affect the model parameters. There were several error flags in the values of data set which were replaced with interpolation as specific information of the conditions were not available.

To develop the forecast model, the sample data were passed through a low-pass filter to achieve better generalization during the training stage. This was done to simplify implementation of the neural network's training. Apparently a low-pass filter captures more of the trend of the series and removes rapid shocks. Since the model is developed for short term (hourly) wind speed forecast, removing rapid changes might result in omitting important information. This was evident during the simulations; the network outcomes were better when hourly changes in observed wind speed closely resembled the filtered series. This problem can be overcome by improving filtering techniques. Implementation of band splitting filter, Kalman filter, Wavelet transformations etc. with the developed model can be some possible solutions to improve the scenario. Additionally, further analysis of season-wise and month-wise wind behavior of the location will be required to improve general performance of the model over more widespread range of dataset.

One of the disadvantages of applying a filter is that, it is almost impossible to reconstruct the predictions in the exact same domain. An easier solution to this can be differencing. Differencing is typically used to make a series stationary by removing trends. It is easy to reverse the differenced series simply by addition. With these particular data used for this project, differencing was not much helpful.

No explicit guideline could be found on determining the number of hidden neurons. To keep the network simple and stable, the model is developed with low number of hidden neurons by trial and error. One convention states that the number of hidden neurons can be chosen as log(T), where T is the number of time instance. But sufficient supporting evidence was not found to take this convention as a hard and fast rule. Some complex algorithms are possible to be implemented to choose the hidden layer size as well as optimal delays for the dynamic network for more efficient forecasting. These are subject for further detailed study. More appropriate combination of hidden layer size, weights and biases and subsets for optimal delays for this particular problem could be different than the used values.

6.3 Future Work:

There are multiple opportunities for future work with the developed NARNET model for hourly wind speed forecast. NARNET architecture is capable to perform multi-step ahead forecast by implementing close loop structure which is an advantage over general ARMA, ARIMA or other linear models. The closed loop basically provides error feedback to the hidden layer to generate forecast of the next point. Higher error margin precision is required in the open loop training to achieve desired close loop goals, hence more accurate multistep ahead forecast results. The methods discussed in the previous section can be helpful in that regard. However, in this study, the multistep-ahead simulations were performed by creating a manual loop to feed the forecast result and layer states back to the input.

Another neural network architecture quite similar to NARNET is nonlinear autoregressive neural network with exogenous inputs (NARXNET). This architecture can use multiple corelated variables, for example wind speed, solar radiation etc. to forecast multi-step ahead wind speed. Also, power generation can be forecasted in this manner. A wind power generation forecast model can also be developed using NARNET depending on availability of power generation data. In several studies it was found that statistical hybrid methods along with physical methods can provide higher accuracy of forecast. The study to develop the NARNET model for step-ahead wind speed forecast can be a good starting point for these future work possibilities.

References

1. http://www.gwec.net/wp-content/uploads/2012/06/Global-Cumulative-Installed-Wind-Capacity-2001-2016.jpg

2. http://www.gwec.net/wp-content/uploads/2012/06/Global-Cumulative-Installed-Wind-Capacity-2001-2016.jpg

3. American wind power reaches major power generation milestones in 2013: American Wind Energy Association (AWEA) press release, March 05, 2014 http://www.awea.org/MediaCenter/pressrelease.aspx?ItemNumber=6184

4. U.S. Energy Information Administration (www.eia.gov) https://www.eia.gov/todayinenergy/detail.php?id=31032#tab1

5. https://en.wikipedia.org/wiki/Growth_of_wind_power_in_the_United_States#cite_ref-9

6. Zhang Y., Wang J., Wang X., "Review on Probabilistic Forecasting of Wind Power Generation", Renewable and Sustainable Energy Reviews, vol. 32, 2014, pp 255-270,.

7. Costa A., Crespo A., Navarro J., Lizcano G., Madesn H., Feitosa E., "A Review on the Young History of Wind Power Short Term Prediction", Renewable and Sustainable Energy Reviews, vol. 12, issue 6, August 2008, pp. 1725-1744

8. Wang X., Guo P., Huang X., "A Review of Wind Power Forecasting Models", Energy Procedia, ICSGCE 2001, Chengdu, China, September 2001

9. Razusi P.C., Eremia M., "Prediction of Wind Power by Artificial Intelligence Techniques", Intelligent System Application to Power Systems (ISAP), 16th International Conference, September 2011

10. Soman S., Zareipour H., Malik O., Mandal P., "A review of Wind Power and Wind Speed Forecasting Methods With Different Time Horizons", North American Power Symposium (NAPS), September 2010

11. National Oceanic and Atmospheric Administration (NOAA), www.noaa.gov

12. http://web.mit.edu/windenergy/windweek/Presentations/Brower_MIT_Wind_Workshop.pdf

13. Lei M., Shiyan L., Chuanwen J., Liu H., Yan Z., "A review on the forecasting of wind speed and generated power", Renewable & Sustainable Energy Reviews, 13 (2009) 915-920, ELSEVIER

14. Catalao J P S, Pousinho H M I, Mendes V M F, "An artificial neural network approach for short-term wind power forecasting in Portugal", Engineering Intelligent Systems, vol 17 no. 1, pp 5 -11, March 2009

15. Potter C, Ringrose M, Negnevitsky M, "Short Term Wind Forecasting Techniques for Power Generation" Australian Universities Power Engineering Conference (AUPEC 2004), Brisbane, Australia, September 2004 https://www.researchgate.net/publication/228870537_Short-term_wind_forecasting_techniques_for_power_generation

16. Costa A., Crespo A., Navarro J., Lizcano G., Madesn H., Feitosa E., "A review on the young history of the wind power short-term prediction", Renewable and Sustainable Energy Reviews 12 (2008) 1725 – 1744, ELSEVIER, January 2007

17. M.G.De Giorgi, A. Ficarella, M.G. Russo, "Short-term Wind Forecasting Using Artificial Neural Networks (ANNs)", WIT Transactions on Ecology and the Environment, vol. 121, 2009, WIT Press, ISSN 1743-3541

18. Erasmo Cadenas, Wilfrido Rivera, Rafael Campos-Amezcua, Cristopher Heard, "Wind Speed Prediction Using a Univariate ARIMA Model and a Multivariate NARX Model", Energies 2016, 9, 109, MDPI. doi: 10.3390/en9020109

19. Michael Brower, "Wind Energy Forecasting", AWS Truepower presentation, January 2011 http://web.mit.edu/windenergy/windweek/Presentations/Brower_MIT_Wind_Workshop.pdf

20. Allan F. Murray, "Applications of Neural Networks", ch. 1, ISBN. 0-7923-9442-9

21. Erasmo Cadenas, Wilfrido Rivera, "Wind Speed Forecasting in Three Regions of Mexico, Using a Hybrid ARIMA-ANN Model", Renewable Energy 35(2010), pp. 2732-2738, ELSEVIER, May 2010

22. Catalao J P S, Mariano S J P S, Mendes V F M, Ferreira L A F M, "An Artificial Neural Network Approach for Short Term Electricity Prices Forecasting", Intelligent System Electrical Eng. Commun., 15(1), pp 15-23

23. https://www.mathworks.com/discovery/supervised-learning.html

24. https://www.mathworks.com/discovery/unsupervised-learning.html

25. J.P.S. Catalao, S.J.P.S. Mariano, V.F.M.Mendes, L.A.F.M. Ferreira, "An Artificial Neural Network Approach for Short-Term Electricity Prices Forecasting", 2007 International Conference on Intelligent Systems Applications to Power Systems, Toki Messe, Niigata, 2007, pp. 1-6. doi: 10.1109/ISAP.2007.4441655

Appendix A - MATLAB Code

Main:

```
clear all; clc; rng(5)

load('date_speed_hour1');
%a= Hourly speeds for 7 years. size(M) = 61368

fct_horizon = 8 % Steps ahead0, hours
% fd = user input from ACF observation
hidden = 4;
ip_data = 744; % Number of input data points
ending = 870;
starting = ending - ip_data + 1;
data_series = starting:ending;
ind1 = ip_data-100; ind2 = ip_data; % ind1, ind2 are plot indices


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% Define sample data, check statistics %%%%%%%%%%%%%%%
A1 = a(data_series)'; % Original comparison set, not normalized
[Max_all, Min_all, Mean_all, SD_all, Var_coeff_all] = statistical(A1);
W2 = A1(ip_data-336:ip_data);
[Max_2W, Min_2W, Mean_2W, SD_2W, Var_coeff_2W] = statistical(W2);
D2 = A1(ip_data-48:ip_data);
[Max_2D, Min_2D, Mean_2D, SD_2D, Var_coeff_2D] = statistical(D2);
display('Statistical properties of observed data');
T1 = table([Max_all; Min_all; Mean_all; SD_all; Var_coeff_all],...
    [Max_2W; Min_2W; Mean_2W; SD_2W; Var_coeff_2W],...
    [Max_2D; Min_2D; Mean_2D; SD_2D; Var_coeff_2D],...
    'VariableNames',{'All_points', 'Last_2_weeks', 'Last_2_days'},...
    'RowNames',{'Max','Min', 'Mean', 'SD', 'Var_coeff'})


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Applying lowpass filter on the sample, check statistics %%
d = fdesign.lowpass('Fp,Fst,Ap,Ast',3,4,0.5,50,10);
Hd = design(d, 'equiripple');
A = filtfilt(Hd.Numerator,1,A1);
b = filtfilt(Hd.Numerator,1,a);

[Max_all_f, Min_all_f, Mean_all_f, SD_all_f, Var_coeff_all_f] = ...
    statistical(A);
W2_filt = A(ip_data-336:ip_data);
[Max_2W_f, Min_2W_f, Mean_2W_f, SD_2W_f, Var_coeff_2W_f] = ...
    statistical(W2_filt);
D2_filt = A(ip_data-48:ip_data);
[Max_2D_f, Min_2D_f, Mean_2D_f, SD_2D_f, Var_coeff_2D_f] = ...
    statistical(D2_filt);
display('Statistical properties of filtered data');
```

```
T2 = table([Max_all_f; Min_all_f; Mean_all_f; SD_all_f; ...
    Var_coeff_all_f],[Max_2W_f; Min_2W_f; Mean_2W_f; SD_2W_f;...
    Var_coeff_2W_f], [Max_2D_f; Min_2D_f; Mean_2D_f; SD_2D_f;...
    Var_coeff_2D_f], 'VariableNames',{'All_points', 'Last_2_weeks',...
    'Last_2_days'}, 'RowNames',{'Max','Min', 'Mean', 'SD', 'Var_coeff'})


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot ACF function of the of the original and filtered series

while(1)
    prompt1 = ('Press any number to start the forecast model, or press 0 to
quit ');
    quit = input(prompt1);
    if quit == 0;
        clc
        break;
    end

    figure(1)
    plot(1:100, A1(1:100), 1:100, A(1:100));
    title('Observed series vs filtered series');
    legend('Observed series', 'Filtered series');

    figure(2);
    [acf, alags, abounds] = autocorr(A, length(A)-1);
    bar(acf, 'b');
    grid on; grid minor;
    title('ACF plot');
    xlabel('Lags');
    ylabel('ACF');
    hold on;
    plot(xlim,[0.2 0.2], 'r', xlim, [-0.2 -0.2], 'r');
    axis([-5 100 -0.8 1.2])

    figure(3)
    autocorr(A, 100); grid on; grid minor;

    prompt2 = 'Please enter optimal lag number from ACF observation: ';
    fd = input(prompt2);
    if fd <1
        disp('Lag should be greater than or equal to 2, please restart the
program');
        break;
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%% Normalizing and NN preparation %%%%%%%%%%%%%
    p = (A-min(A))/(max(A)-min(A)); % Comparison set, normalized
    p = con2seq(p);
    t = p; % Target set
    t_new = t;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% Network parameters %%%%%%%%%%%%%%%%%%%%
```

```
    net = narnet(1:fd, hidden, 'open', 'trainlm');
    net.inputs{1}.processFcns = {};
    net.outputs{1}.processFcns = {};
    net.inputWeights{1}.initFcn = 'randsmall';
    net.biases{1}.initFcn = 'randsmall';
    net.biases{2}.initFcn = 'randsmall';

    net.divideFcn = 'divideblock';
    net.performParam.regularization = 10^-5;
    net.performFcn = 'MSE';
    net.trainParam.goal = 1e-10;
    net.trainParam.epochs = 10000;
    net.trainParam.show = 10;
    net.trainParam.max_fail = 6;
    net.layers{1}.transferFcn = 'tansig';
    net.layers{2}.transferFcn = 'purelin';

    net.trainParam.mu = 0.05;
    net.trainParam.mu_dec = 0.8;
    net.trainParam.mu_inc = 1.1;
    net.trainParam.showWindow = true;

    net.divideParam.trainRatio = 0.7;
    net.divideParam.valRatio = 0.15;
    net.divideParam.testRatio = 0.15;

    out1 = zeros(1, fct_horizon);
    trn_ind = 1: floor(0.7*(length(t)-fd));
    val_ind = trn_ind(end)+1:(trn_ind(end) + floor(0.15*(length(t)-fd)));
    tst_ind = val_ind(end)+1:(val_ind(end) + floor(0.15*(length(t)-fd)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for m = 1:fct_horizon
        t = t_new;
        [Xs_o, Xi_o, Ai_o, Ts_o, EWs_o, shift_o] = preparets(net, {},...
            {}, t);
        net = train(net, Xs_o, Ts_o, Xi_o, Ai_o);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Open loop performance
        [yo, Xfo, Afo] = net(Xs_o, Xi_o, Ai_o);
        ts_o = cell2mat(Ts_o);
        ys_o = cell2mat(yo);
        perf_open_training = perform(net, yo, Ts_o)/var(ts_o, 1) %MSE overall
%%%%%%%%%%%%%%%%%%%% Open loop training ends %%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%% One step ahead %%%%%%%%%%%%%%%%%%%%%%%
        nets = removedelay(net);
        [Xs, Xis, Ais, Ts] = preparets(nets, {}, {}, t);
        Ypred = nets(Xs, Xis, Ais);
        Ypred = cell2mat(Ypred);
        Ypred = Ypred(length(Ypred))*(max(A)-min(A))+min(A);
        out1(m) = abs(Ypred);
        t=cell2mat(t);
        t_new = t;
        for n = 2:length(t)
            t_new(n-1) = t(n);
        end
        t_new(end) = Ypred/max(A);
        t_new = con2seq(t_new);
    end
    Y_real = b((starting+ip_data):(starting+ip_data+fct_horizon-1))';
    Error = (abs(Y_real-out1))./Y_real;
    Error = Error*100


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%% Plot open loop training result %%%%%%%%%%%%%%%%%%%%%%
yon = reconstruct_unnormalize(A, cell2mat(yo));
Ts_on = reconstruct_unnormalize(A, cell2mat(Ts_o));
    figure(4);
    plot(fd+1:length(t), Ts_on, 'b', fd+1:length(t), yon, 'r--');
    title('Target vs prediction');
    xlabel('Time, hour'); ylabel('Wind Speed, m/s');
    legend('Target', 'Prediction');
    axis([ind1 ind2 min(a)-1 max(a)+1]);

%%%%%%%%%%%%%%%%%%% Plot forecast result %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(5);
    plot(1:fct_horizon, Y_real, 1:fct_horizon, out1);
    title('Forecast results');
    xlabel('Time, hour'); ylabel('Wind Speed, m/s');
    legend('Observed wind speed ', 'Forecasted wind speed');
    grid on;
    axis([1 10 0 (max(max(Y_real),max(out1))+2)]);

%%%%%%%%%%%%%%%%%%% Plot Error %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(6);
    plot(1:fct_horizon, Error);
    title('Error(%) vs Time');
    xlabel('Time, hour'); ylabel('Error(%)');
    break;
end


[MAE_train, MSE_train, MPE_train, MAPE_train] = Error_stat...
    (ts_o(trn_ind), ys_o(trn_ind));
[MAE_test, MSE_test, MPE_test, MAPE_test] = Error_stat...
    (ts_o(tst_ind), ys_o(tst_ind));
[MAE_val, MSE_val, MPE_val, MAPE_val] = Error_stat...
    (ts_o(val_ind), ys_o(val_ind));
ps_o = ts_o;
ps_y= zeros(length(ps_o), 1);
for i = 1:(length(ts_o)-1)
    ps_y(i+1) = ps_o(i);
end
```

```
Actual= [a(ending+1) a(ending+2) a(ending+3) a(ending+4) a(ending+5)...
    a(ending+6) a(ending+7) a(ending+8)]
Filtered = [b(ending+1) b(ending+2) b(ending+3) b(ending+4) ...
    b(ending+5) b(ending+6) b(ending+7) b(ending+8)]
Forecast = out1
error = abs(Filtered - Forecast);
for i = 1:length(Filtered)
    error(i) = (error(i)/Filtered(i))*100;
end
error'
MSE_test
```

Function 1:

```
function [Max, Min, Mean, SD, Var_coeff] = statistical(A)

Max = max(A); Min = min(A); Mean = mean(A); SD = std(A);
q = zeros(round(max(A)), 1);
%Variance = SD^2;

for i = 1:round(max(A))
    q(i) = length(find((A>(i-1)) & (A<=i)));
end
l = max(q);
% Mode = find(q==l);

Var_coeff = (SD/Mean)*100;
```

Function 2:

```
function series = reconstruct_unnormalize(original_series, normalized_series)

series = zeros(1, length(normalized_series));

for i = 1:length(normalized_series)
    series(i) = ((normalized_series(i))*(max(original_series) -…
min(original_series))) + min(original_series);
end
```

Function 3:

```
function [MAE, MSE, MPE, MAPE] = Error_stat(predicted_series,
observed_series)

sum = 0; sum2 = 0; sum3 = 0; sum4= 0;
err = 0; err2 = 0; err3 = 0; err4 = 0;

for i = 1:length(predicted_series)
    err = abs(predicted_series(i) - observed_series(i));
    sum = sum + err;
    err2 = err^2;
    sum2 = sum2 + err2;
    if(observed_series(i)== 0)
        err3 = (observed_series(i) - predicted_series(i))*100;
    else
        err3 = ((observed_series(i) -
predicted_series(i))/observed_series(i))*100;
    end
    sum3 = sum3 + err3;
    err4 = abs(err3);
    sum4 = err4 + sum4;

end
MAE = sum/length(predicted_series);
MSE = sum2/length(predicted_series);
MPE = sum3/length(predicted_series);
MAPE = sum4/length(predicted_series);
```