JACOBS
UNIVERSITY

Web Traffic Prediction using Echo State Networks

by

Lalit Singh

Bachelor Thesis in Computer Science

Prof. Dr. Herbert Jaeger
Bachelor Thesis Supervisor

Date of Submission: May 12, 2018

Jacobs University — Focus Area Mobility

Abstract

Information about the potential web traffic that any particular website might receive in future is essential for managing the server resources of the website. Online advertisers rely on the predicted web traffic for making their advertisement related decisions. Therefore, they have to anticipate the web traffic using various methods. Methods that have been previously used for this task are computationally expensive and have multiple problems. Echo State Networks (ESN) are novel type of ANN that has achieved good results in a variety of tasks with a computationally cheaper algorithm. This paper presents how ESN can be used for web traffic prediction task. The are many parameters that need to be tuned and various data preprocessing techniques that need to be followed to obtain good results for the prediction. This paper explains how this is done.

Contents

1 Introduction

1.1 Motivation of Research

The ability to predict the future data, based on past data, makes an important leverage that can push the organization forward. Time series forecasting is an important tool under this scenario where the goal is to predict the behaviour of complex systems solely by looking at patterns in past data. A time series is a collection of periodic ordered observations that appear in multiple ranges of domains such as literature, agriculture, finance, media, etc., just to name a few [2]. Time series are very complex because each observation is dependent upon the previous observations and is often influenced by more than one previous observations [20]. Hence, this research mainly focuses on the prediction of time series data in web traffic domain.
Information about the potential web traffic that a website might receive in future is essential to the management of server resources and potentially prevent the Denial of Service (DoS) during peak hours. Online advertising business also depends upon the predicted web traffic because they need to distribute advertisements to the selected websites based on the predicted web traffic [16]. As the estimation of web traffic is important due to aforementioned purposes, several research works have been done in this sector.

1.2 Prior Works

Traditionally these time series forecasting have been performed using various statistical-based methods [15]. The major drawback of most of these statistical models is that they consider the time series are generated from a linear process. However, most of the real world time series generated often consists of temporal and/or spatial variability and suffers from nonlinearity of underlying data generating process [17]. In [16], the author used both MLP and RNN for the prediction of web traffic and found out that MLP outperformed RNN. Despite extensive experimentation which involved tuning various parameters, the predictions produced by RNN were consistently poor in terms of accuracy [16]. Work in [1] used Neural Network to forecast prices of flour in three different cities. They compared their results with autoregressive moving average (ARMA) model and found that Neural Network approach outperformed the ARMA model.

In [21], the author uses Recurrent Neural Network sequence to sequence model where he used data features such as days of the week, year to year autocorrelation, quarter to quarter autocorrelation and page popularity to train his model. His model outperformed all the model in a web traffic forecasting competition organised by Google on Kaggle (a platform for predictive modelling and analytics competitions). Another participant of the same competition used Kalman filter to make the prediction [19].

All the methods that have been used before have some sort of disadvantages. ANNs require a complex training process, they may converge to a local minimum, they have difficulty in determining the optimum network structure, and they experience fading memory (FM) [12]. Therefore, it is difficult to create a more accurate web traffic prediction model using ANNs. Echo state networks (ESNs) are a novel type of ANN proposed by Jaeger in 2001 and are regarded as the closest representation of the learning process of the human brain [8]. ESNs can effectively solve some of the aforementioned problems with

ANNs. In addition, the computational requirement for training ESN is lesser than training ANNs and MLPs because while training ESN the connection weights of the reservoir are not changed and only weights from the reservoir to the output units are adapted, so training becomes a linear regression task [4, 12]. Therefore, we have chosen to use ESNs for the prediction of web traffic.

1.3 Research Objective

Firstly, I would like to state that the goal of this project is to neither compete with the state of the art solution nor to compare the results with those solutions. Rather, it is to see the possibility of the use of Echo State Network for web traffic forecasting tasks. The primary objectives for conducting this research are listed below:

- To evaluate the performance of ESN for the prediction of web traffic.
- To optimize the parameters of ESN for its best performance.
- To find out what additional signal helps to improve the prediction task by ESN.

2 Background Concepts

2.1 Echo State Networks

2.2 Introduction

Echo State Networks (ESNs) are a special kind of Recurrent Neural Networks(RNNs), with an easy to implement training algorithm. Large, sparsely connected RNN with N neurons is used as a "Dynamic Reservoir". Each neuron in this reservoir is connected to each other with certain weights. The basic idea of ESN is (i) to drive this network of neurons with K input signals, thereby inducing in each neuron within this reservoir network a nonlinear response signal, and (ii) combine a desired output signal by a trainable linear combination of all of these response signals [7]. The reservoir can be seen as a nonlinear high dimensional expansion of input signal which serves as a memory providing temporal context. Therefore, the reservoir being an input-driven dynamical system should provide a rich and relevant enough signal space such that desired signal can be obtained by linear combination of it [13]. Under suitable conditions, the network state becomes asymptotically independent of initial conditions and depends only on the input history, which is called the "Echo State Property". This means that all desired output signals can be built out of its own "echos" in the Dynamic Reservoir [5].

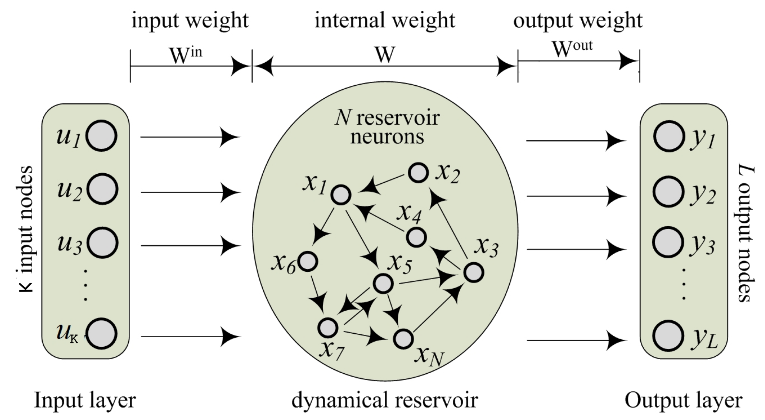


Figure 1: The basic schema of Echo State Network. Image source[11].

2.3 Formalism

2.3.1 Formulation of problem

Let a *problem* or a *task* in our context of machine learning be defined as a problem of learning a functional relation between a given input $\mathbf{u}(n) \in \mathbb{R}^{Nu}$ and a desired output $\mathbf{y}^{\text{target}}(n) \in \mathbb{R}^{Ny}$. Our goal is to learn a function $\mathbf{y}(n) = y(\mathbf{u}(n))$ such that $E(\mathbf{y}, \mathbf{y}^{target})$ is minimized where E is the training error measure, for example, normalized root-mean-square error (NRMSE) (6) [14].

2.3.2 Theory

We consider discrete time neural networks with K input units, N internal network units and L output units. Each of units at time step n has activation. Activations of input units at time step n are $\mathbf{u}(n) = (u_1(n), \dots, u_K(n))$, of internal units are $\mathbf{x}(n) = (x_1(n), \dots, x_N(n))$ and of output units are $\mathbf{y}(n) = (y_1(n), \dots, y_L(n))$. Real valued connection weights for the input links are collected in the matrix $\mathbf{W}^{in} \in \mathbb{R}^{N\times K}$, for the synaptic links between the neurons within the network are collected in matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$, and for the output weights are collected in matrix $\mathbf{W}^{out} \in \mathbb{R}^{L\times(K+N)}$ [10]. The connections from input to output are used and the corresponding connection weights are stored in matrix $\mathbf{W}^{out}$. The output units may optionally project back to internal units with connections however they have not been used in this experiment. A zero weight value can be interpreted as "no connection" [6].
The activation of internal units are updated according to the discrete time state update equation (1).

$$\mathbf{x}(n+1) = f(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{in}\mathbf{u}(n+1) + \mathbf{B}) \tag{1}$$

where $\mathbf{u}(n)$ is the externally given input,$f()$ denotes the component-wise application of unit output function ($tanh$ used for this experiment), and $\mathbf{B} \in \mathbb{R}^N$ is the bias vector.

The extended system state **z**(n) = [**x**(n) ; **u**(n)] is obtained by the concatenation of system states and input signal at time $\mathbf{n}$. For each time point, $i = 1, \dots, n_{max}$, the extended system state is stacked row-wise in a state collection matrix **S**. This process is known as harvesting reservoir state. The output is obtained from the extended system states using Equation (2).

$$\mathbf{y}(n) = \mathbf{g}(\mathbf{W}^{out}\mathbf{z}(n)) \tag{2}$$

where $\mathbf{g} = (g_1, \ldots, g_L)$ is unit output activation function (in our case identity).[7].

The **W**^{out} matrix consisting the weights of links connecting reservoirs to output nodes is obtained by the linear regression of weights of desired outputs **d**(n) on the harvested extended systems states **z**(n). **W**^{out} can be computed efficiently by linear regression of **S** and **D** as given by Equation (3).

$$\mathbf{W}^{out} = (\mathbf{S}^{'}\mathbf{S} + \alpha\mathbf{I})^{-1}\mathbf{S}\mathbf{D}^{'} \tag{3}$$

where α is the regularization coefficient **D** is desired output vector[7].

3 Description of the Investigation

3.1 Data Description and Preprocessing

The dataset used for this experiment was obtained from Kaggle (a platform for predictive modelling and analytics competitions). The data set was provided by Google for a web traffic prediction competition. This data file consists of a number of views for 145063 Wikipedia pages on each date starting from 1st of July 2015 to 31st of December 2016. Each time series consists of data for the traffic generated by humans, bots or both from devices like mobile or desktop. For each date, the data contains a single integer value representing the number of views that the particular Wikipedia page received on that date. The total length of each time series is 550. There are some missing data points for some of the time series. In this experiment, we have ignored such time series for the sake of convenience. Out of 145063 time series, for this study, we have chosen the traffic generated for the page named "2002 FIFA World Cup" on English version of Wikipedia (i.e `https://en.wikipedia.org/`) on desktop devices by all types of agents including humans and bots.

For the preprocessing of the time-series used for this experiment, following steps are followed:

Step 1

The time series data used in this experiment has very large values (≈ 7000) as seen in Figure 2. Therefore, the time-series data is squeezed componentwise such that the maximum value, MAX of the time-series is squeezed to the new maximum value max. This is done by taking small ($<< 1$) power of each component in time series $\mathrm{s}(n) = (s(1), \ldots, s(n))$. The power p that needs to be raised to each component of time-series is

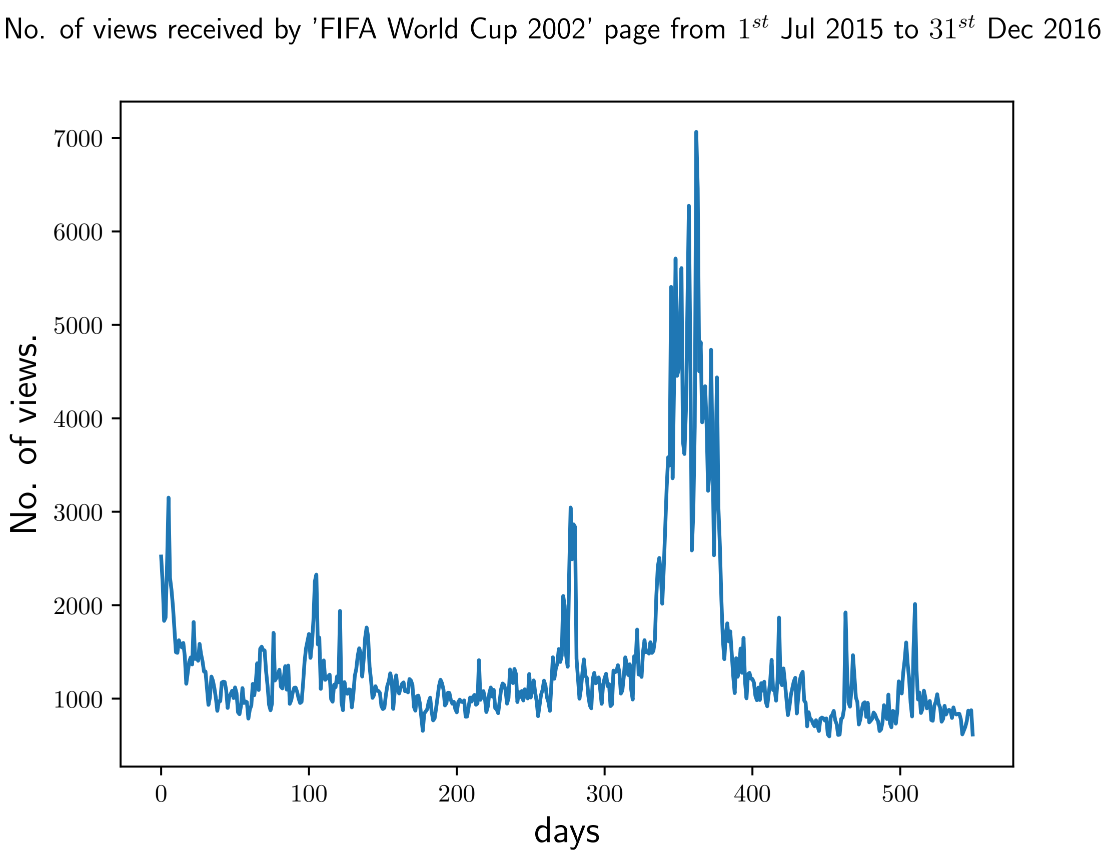
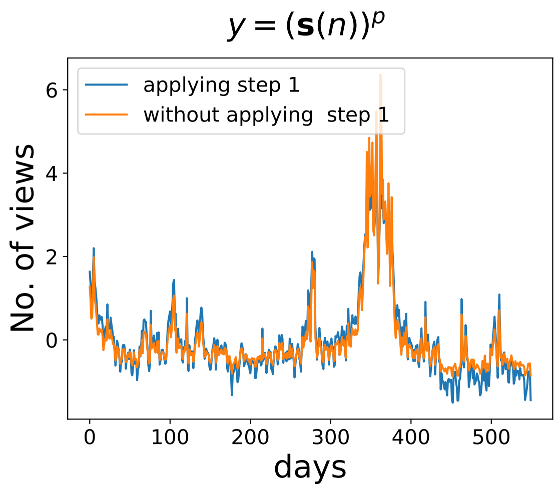
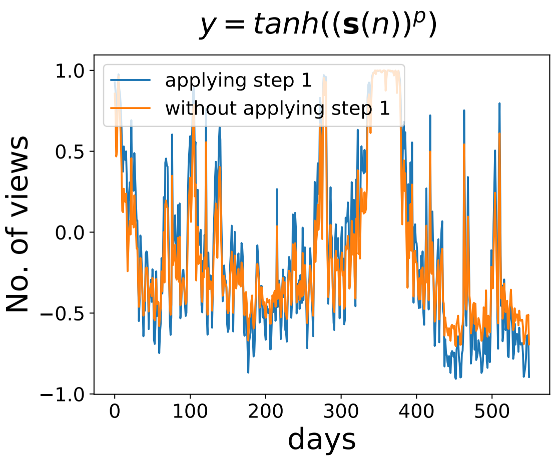


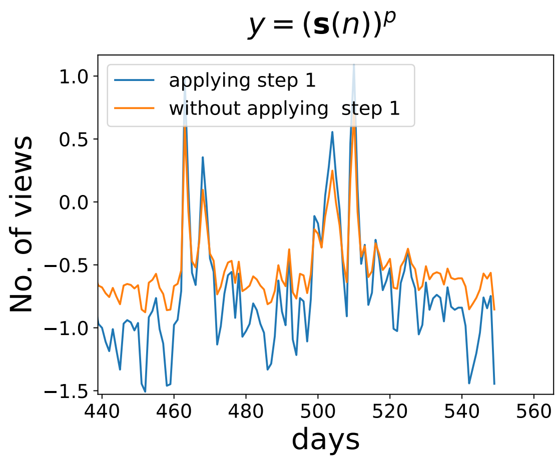
Figure 2: Raw data for Wikipedia Page "2002 FIFA World Cup" which has number of views of on each date from 1st of July 2015 to 31st of December 2016.



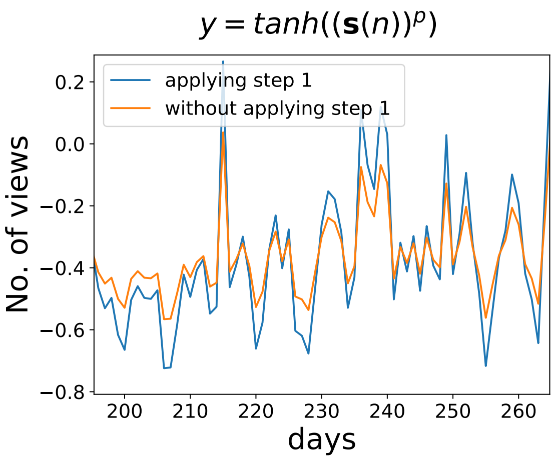
(a) After taking component-wise power



(b) After further taking the $tanh$ of the signal



(c) After further taking the $tanh$ of the signal



(d) After further taking the $tanh$ of the signal

Figure 3: Preprocessing steps

calculated using Eq. 4. The value of max can be optimized for the best performance of the network.

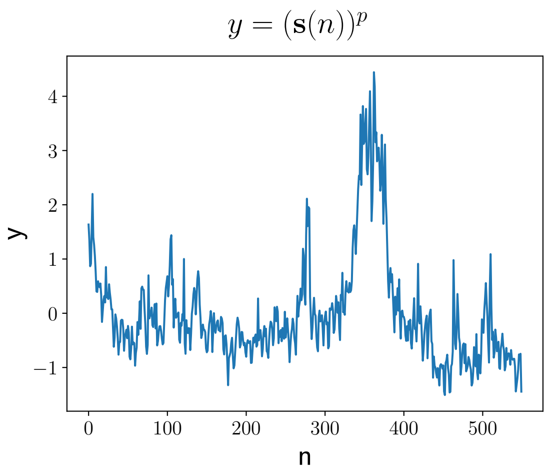
$$\begin{aligned} max &= MAX^p \\ p &= log_{MAX}(max) \end{aligned} \tag{4}$$

Step 2

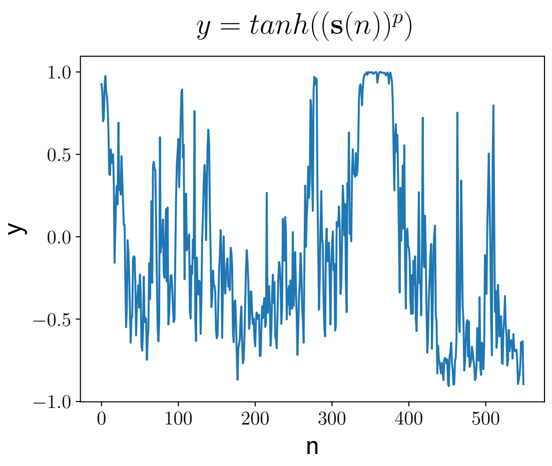
The resultant time series from step 1 is further rescaled and shifted such that it has zero mean and unit variance. The time-series data is divided componentwise by its standard deviation to make its variance unit.

Step 3

The resultant time series from step 2 is further applied a $tanh$ function componentwise. This strictly scales the data points in time series in the range of -1 to 1.



(a) After taking component-wise power



(b) After further taking the $tanh$ of the signal

Figure 4: Preprocessing steps

In step 1, the higher valued data points in the signal are penalised more than the smaller ones. Therefore, the higher peaks in the input signal are shortened, and the smaller wiggles in the signal are scaled up. In Fig. 3 a (a zoomed in version in Fig. 3 c) the yellow signal generated with the application of only step 2 has lesser wiggleness than the blue signal generated with the application of both step 1 and step 2. Even after the application of $tanh$ in step 3, the wiggleness in the blue signal generated by applying all three steps is higher than the yellow signal that is generated without applying step 1 (see Fig. 3 a, or a zoomed in version in 3 c). Patterns become more prominent for ESN with the increase in these wiggleness. Applying $tanh$ function in step 3 further scales up the smaller wiggles and scales down the higher peaks such that the values at any time point in signal is within the range of -1 to 1.

3.2 Additional Input Signals

By nature, the number of views that a particular Wikipedia Page receives vary according to days in week and months in a year. For instance: A page related to weekend activities might receive a different proportion of views on weekends than on working days. Further, Wikipedia pages on related subject matters might receive proportional number of views and could follow similar pattern of incoming web traffic. Therefore those related input signals were added to the ESN to improvise the prediction result. In particular, we used the following approaches to find suitable additional input for the network.

1. First of all, we calculate a sine wave that has the highest correlation with the main input signal. It gives the insights about the weekly pattern to the ESN. It was further preprocessed using the same routines as used for the main input signal. This sine

wave signal was further scaled with a scaling factor sf_{sine} and then input to the network.

2. Then from the database of about 145 thousand time series top 20 time series are selected which have the highest correlation with the main input signal. The main input signal of page "2002 FIFA World Cup" on English version of Wikipedia i.e `https://en.wikipedia.org/` has high correlation to the pages given in Table 1. The main input signal and the mean of these additional signal after tuning it to zero mean and unit variation is plotted in Fig 5. As the additional signals are highly correlated to the main signal, these two signal has similar patterns in longer range.

Table 1: Information about the additional signal

Wikipedia page Title	**Wikipedia version**	**correlation coefficient**
Fußball Weltmeisterschaft 1994	de.wikipedia.org	0.92
2010 FIFA World Cup	en.wikipedia.org	0.94
Copa Mundial de Fútbol de 2010	es.wikipedia.org	0.91
Fußball-Europameisterschaft 2008	de.wikipedia.org	0.90
Fußball-Europameisterschaft 1992	de.wikipedia.org	0.91
UEFA Euro 2008	en.wikipedia.org	0.91
Eurocopa 2008	es.wikipedia.org	0.93
2006 FIFA World Cup	en.wikipedia.org	0.91
Fußball-Weltmeisterschaft 2010	de.wikipedia.org	0.92
2010 FIFA World Cup	en.wikipedia.org	0.93
1998 FIFA World Cup	en.wikipedia.org	0.96
Coupe du monde de football	fr.wikipedia.org	0.91
Fußball-Europameisterschaft 1988	de.wikipedia.org	0.92
2006 FIFA World Cup	en.wikipedia.org	0.96
Чемпионат мира по футболу 2002	ru.wikipedia.org	0.90
Fußball-Weltmeisterschaft 2002	de.wikipedia.org	0.93
2002 FIFA World Cup	en.wikipedia.org	0.96
Fußball-Weltmeisterschaft 2014	de.wikipedia.org	0.91
2014 FIFA World Cup	en.wikipedia.org	0.91
Чемпионат мира по футболу 2010	ru.wikipedia.org	0.92

These signals are further preprocessed using the same preprocessing routines as used for preprocessing the main input signal. Finally, each of this additional signals is multiplied with a scaling factor sf_{add}/NUM, where NUM is the total number of such additional inputs. To determine the correlation between any two time series we have used Pearson's correlation coefficient.

3.2.1 Pearson correlation coefficient

Pearson's correlation coefficient is the measure of the linear correlation coefficient between two time series. Its value is in the range of -1 to 1. Value close to 1 indicates that the time series are highly correlated to each other, value close -1 indicates that the time series are negatively correlated to each other and value close to 0 indicates that the two time series are unrelated. Pearson coefficient between to time series $\mathbf{X} = (x_1, \ldots, x_m)$

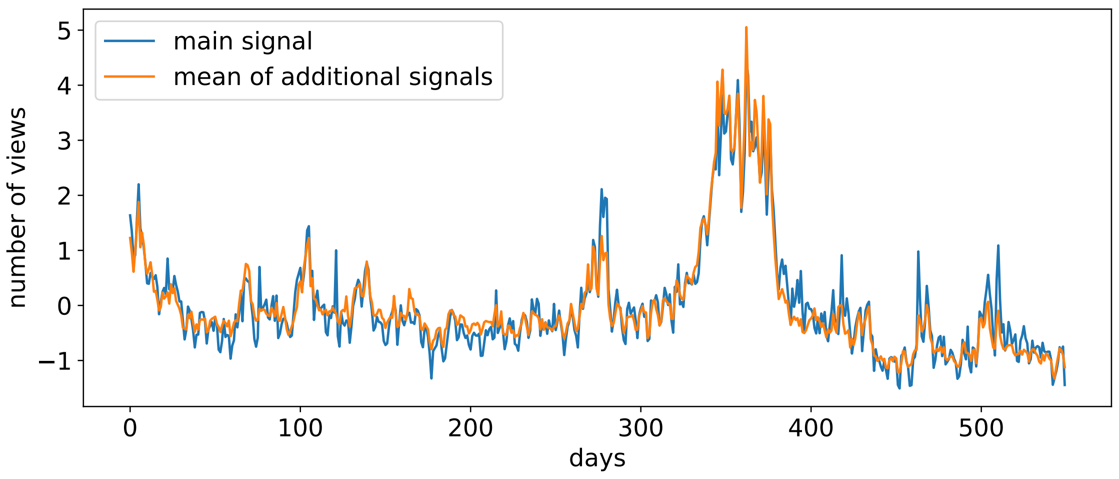


Figure 5: Main signal vs mean signal.

and $\mathbf{Y} = (y_1, \ldots, y_m)$ can be computed using the Equation (5).

$$\begin{aligned}\rho X, Y = &\frac{cov(X, Y)}{\sigma_X \sigma_Y} \\ = &\frac{\Sigma_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma_{i=1}^{m}(x_i - \bar{x}^2)}}\end{aligned} \tag{5}$$

where:

- m is the length of time series
- x_i, y_i are the data points in time series X and Y indexed with i
- $\bar{x}_i = \frac{1}{n}\Sigma_{i=1}^{n} x_i$ and analogously for $\bar{y}$

3.3 Network Setup

We create a reservoir of $N = 99$ neurons. The weights for input links and bias vector are generated randomly from uniform distribution over (-0.5, 0.5) and stored in matrix $\mathbf{W}^{in} \in \mathbb{R}^{N \times K}$ and $\mathbf{B} \in \mathbb{R}^{N \times 1}$ respectively. There are $K = 22$ input neurons that receive 1 main input signal and each of the 21 additional input signals and $L = 31$ output neurons each of which makes $t = 1, \ldots, 31$ step ahead prediction.

In order to achieve good approximation of desired signal, the echo functions should provide a "rich" set of dynamics to combine from. The network should be prepared in a suitably "inhomogeneous" way to meet this demand. One simple method to prepare such a "rich reservoir" echo state network is to supply a network which is sparsely and randomly connected. Sparse connectivity provides for a relative decoupling of subnetworks, which encourages the development of individual dynamics [9]. In this experiment, the network weight matrix, $\mathbf{W}$ has different weight regions to create different capability of

fast	≈ 0	=0
≈ 0	medium	≈ 0
=0	≈ 0	slow

Figure 6: Structure of weight matrix, $\mathbf{W}$.

short term memory so that the network can capture all type of trend (slow, medium and fast trends) in the training signal. The structure of weights matrix used is presented Figure 6.

Values in the weight matrix for in all the regions are drawn from the uniform distribution of range -1 to 1 and are scaled with a scaling factor. The scaling factors for the fast, medium and slow regions are a, b and c respectively such that $a > b > c$. For other regions the scaling factors are almost equal to zero or zero. These scaling factors are optimized during the training period for the better performance of the network.
The spectral radius of the reservoir weight matrix **W** codetermines (i) the effective time constant of the echo state network (larger spectral radius implies slower decay of impulse response) and (ii) the amount of nonlinear interaction of input components through time (larger spectral radius implies longer-range interactions) [7]. The weight matrix $\mathbf{W}_0$ is normalized to $\mathbf{W}_1$ with its spectral radius by putting $\mathbf{W}_1 = |1/\lambda_{max}|\,\mathbf{W}_0$ where λ_{max} is the spectral radius of $\mathbf{W}_0$. Then the weight matrix $\mathbf{W_1}$ is scaled with a scaling factor sf_W such that $\mathbf{W} = sf_W\mathbf{W_1}$. Then $\mathbf{W}$ has spectral radius of sf_W. The choice of the spectral radius sf_W of the reservoir is crucial for the eventual success of ESN training. This is because sf_W is intimately connected to the intrinsic timescale of the dynamics of the reservoir state. Small sf_W means that one has a fast dynamics reservoir and large sf_W (i.e close to unity) means that one has slow reservoir. Also, the input weight matrix $\mathbf{W}^{in}$ and bias vector $\mathbf{B}$ is scaled with a scaling factors sf_Win and sf_B respectively.

3.4 Performance Metrics

We have used Normalized Root Mean Square Error (NRMSE) as a measure to evaluate the performance of the reservoir. NRMSE measures the difference between the predicted values y and true values r. The values for y and r in this experiment are one dimensional. NRMSE for one dimensional y and r can be computed using the Equation (6).

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(y(i) - r(i))^2}{N\sigma^2}} \tag{6}$$

where N denotes the number of output data points, σ^2 denotes the variance of y and r. NRMSE with its value close to zero indicates that the quality of prediction is very good. The higher the NRMSE the lesser accurate the prediction is.

3.5 Regularization

In order to access the quality of the prediction produced by the training of ESN, we regularly monitor the computed output weights $\mathbf{W}^{out}$. Large weights indicate that $\mathbf{W}^{out}$ exploits and amplifies tiny differences among the dimension of $\mathbf{x}(n)$, and can be very sensitive to deviations from the exact conditions in with the network has been trained [13]. To counteract this effect the regularization part $\beta\mathbf{I}$ in the ridge regression is used as in Equation (3) .

3.6 Leaky Integration

In ordinary ESNs, the current state $\mathbf{x}(n+1)$ of the network functionally depends only on the previous state $\mathbf{x}(n)$. This property is excellent for learning discrete chaotic signals with strong oscillations but imposes difficulties when dealing with continuous slow signals. To overcome this problem, we use networks with continuous dynamics. Networks with continuous dynamics can be approximated by updating the network with a leaky integrator. The leaking rate α can be intuitively seen as the value that regulates how much information about the previous state is preserved and how much information is updated by the network update function. When α with its value close to 0 the network state is changed very slowly since only a small 'fraction' of the state is updated. On the other hand, when α with its value close to 1, the network state is updated very quickly. The leaking rate allows Echo State Networks to remember previous states across multiple update cycles allowing the network to learn continuous signals with long periods [10, 18].

The state update Eq. (1) is modified to Eq. (7). The values

$$\mathbf{x}(n+1) = (1-\alpha)\mathbf{x}(n) + \alpha(f(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{in}\mathbf{u}(n+1) + \mathbf{B})) \tag{7}$$

Each component of $\alpha \in \mathbb{R}^{N\times 1}$ can be optimized for the better performance of the network, however, this increases the number of parameters that need to be optimized, with the increase in the size of Network. Therefore to make the tuning process easier the one dimensional vector is divided into three regions so that it has three different leaking rates or scaling factors. The entire vector α consists and the three scaling factor are the parameters that need to be optimized.

3.7 Cross Validation

The length of each available time series data is 550 out of which we leave out the last 31 data points for the testing purpose after the training phase of ESN is completed. Rest of the data points are used for training purpose.
Optimizing a parameter for a reservoir using same set of training and validation data might lead to an overfitting problem. Overfitted model performs well on validation set of data. However, when this overfitted model is tested on new set of testing data, its performance is very poor. Therefore, to avoid such a disaster in performance we have used a concept of cross-validation.
Since the length of the training is only 519 after removing a part of it for the validation part, it poses a risk of underfitting. By reducing the amount of data for training, we risk of not

capturing the important pattern/trends in the data set, which in turn increases the error induced by bias. So we require a cross validation method scheme which leaves ample of data for training and ample of data for validation. In our case, K-fold cross validation is the first choice because it exactly satisfies the previously stated requirement.

3.7.1 K-fold cross validation

In K-fold cross validation, the available signals are split into K parts. Out of those K parts, one of them is used for validation purpose and rest K-1 split sub-signals are concatenated in the original order to form a single signal which is then used for training. This is repeated K times such that all the K splits are used as validation signal exactly once. The overall estimate error of the cross validation, **CV**$_k$ given by Equation (8) , is average for the error in each fold. For this experiment we have used $K = 4$ folds.

$$\mathbf{CV}_k = \frac{1}{k}\sum_{i=1}^{k} NRMSE_i \tag{8}$$

3.7.2 Computational Efficiency in K-fold cross validation

As mentioned in subsection 3.7.1, the input signal is divided into K splits for the cross validation and in each fold of cross validation loop, (K-1) splits are used for training and the remaining as validation signal. However this process of redoing the similar steps multiple times can be computationally redundant and expensive for longer input signal or higher values of K. Therefore, to decrease the computational cost of training the network, instead of splitting input data in K fold and doing cross validation steps, we feed the entire input signal to the state update equation (7) in a loop that runs from $i = 1, \ldots, n_{max}$ (n_{max} is the length of the input signal). In each iteration i of loop, we use state vector **x** of the previous iteration (i.e. at time $i-1$) which goes through a nonlinear transformation to generate state vector at time i. For $i = 0$, we choose the state vector **x** as an N dimensional vector of zeros. These state vectors, $\mathbf{x}(n)$ concatenated with input, $\mathbf{u}(n)$ and are stacked in state collection matrix $\mathbf{S} \in \mathbb{R}^{n_{max}\times(N+k)}$.
As for ESN, before proceeding to the training the network it is necessary to remove the initial few states (eg. zero state) which contains initial memory that is not the part of the input. Therefore first n_0 rows of matrix $\mathbf{S}$ are discarded. The value of n_0 is determined by the observations of the activations of the neurons. By time $n = n_0 + 1$, it is safe to assume that the effects of the arbitrary initial state have washed away and the network gives a pure reflection of the input signal [3].

After removing initial washout from the state collection matrix $\mathbf{S}$, it is divided into K folds. In each iteration of loop that runs from $i = 1, \ldots, K$ the input signal and teacher signal corresponding to one of the set is used as validation data set while the rest of folds and their corresponding teacher signal is used for training the network. This way we only have to compute the states using the entire signal and then reuse those computed states for training and validation during cross validation loop. This procedure reduces the computational cost of training the network during cross validation phase.

3.8 Computation of output weights

The learning of the output weights $\mathbf{W}^{out}$ (3) can be phrased as solving a system of linear equation

$$\mathbf{W}^{out}\mathbf{S} = D \tag{9}$$

with respect to $\mathbf{W}^{out}$. Since the goal is to minimize the quadratic error $E(\mathbf{D}, \mathbf{W}^{out})$ as in (6), to solve (9) we have used methods for finding least square solutions of overdetermined systems of linear equation i.e linear regression [14].

$\mathbf{W}^{out}$ is computed as the linear regression of weights of desired outputs, $\mathbf{D}$, on the harvested extended system states during the training phase **S**. We use Equation (3) to compute **W**^{out}.

4 Results

After the optimization of the variables during cross validation phase of the experiment, the network is retrained using all the available signal using the same setup as in cross validation phase. The remaining portion of the signal excluding the data for last 31 days is used to make the extended system states. These extended systems states and the computed **W**^{out} is used to make prediction using Equation 2.
There are 31 output neurons each of which makes the prediction for $n = 1\ldots, 31$ days ahead. The network optimized during cross validation phase and retrained afterwards is used to predict the number of views for the month of December 2016. The error for each of prediction range is calculated as NRMSE. These error measures are plotted in Figure 7. The predicted number of views and the actual number of views for different prediction horizon are plotted in figure 8. As seen in Figure 7, usually the NRMSE for the smaller prediction horizon is smaller and for the longer prediction horizon the NRMSE is bigger. The NRMSE values do not increase linearly with the prediction horizon. One reason for this could be, I assume, that some of the features like biweekly or triweekly repetition of some pattern in the data might be strong than any other frequency.
The values for the optimizable parameters that worked best for this experiment are presented in the Table 3. Also the mean mean training error and testing error for all 31 prediction horizons are given in the Table 4.

	value
Mean training NRMSE	1.02
Mean Testing NRMSE	1.04

Table 2: Mean training and testing errors of all 31 prediction horizons.

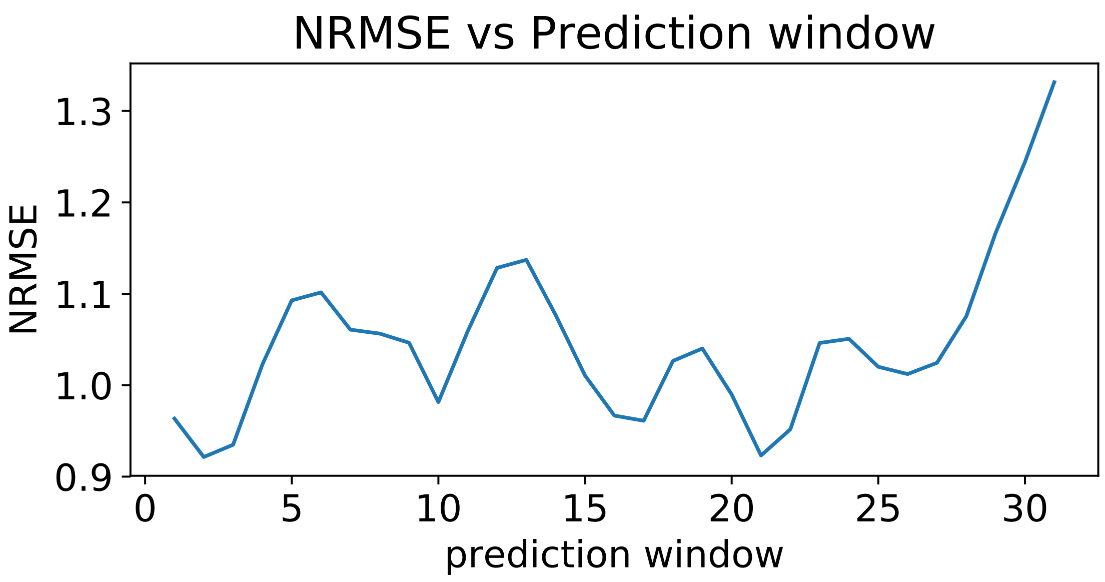


Figure 7: NRMSE for different prediction window ranges

Parameters		Optimized value
$sf^{\mathbf{W}}$		0.9
$sf^{\mathbf{Win}}$		0.3
$sf^{\mathbf{B}}$		0.001
sf_{sine}		0.8
sf_{add}		0.6
α	FAST	0.1
	MEDIUM	0.5
	SLOW	0.9
max		5

Table 3: Optimized values for control parameters

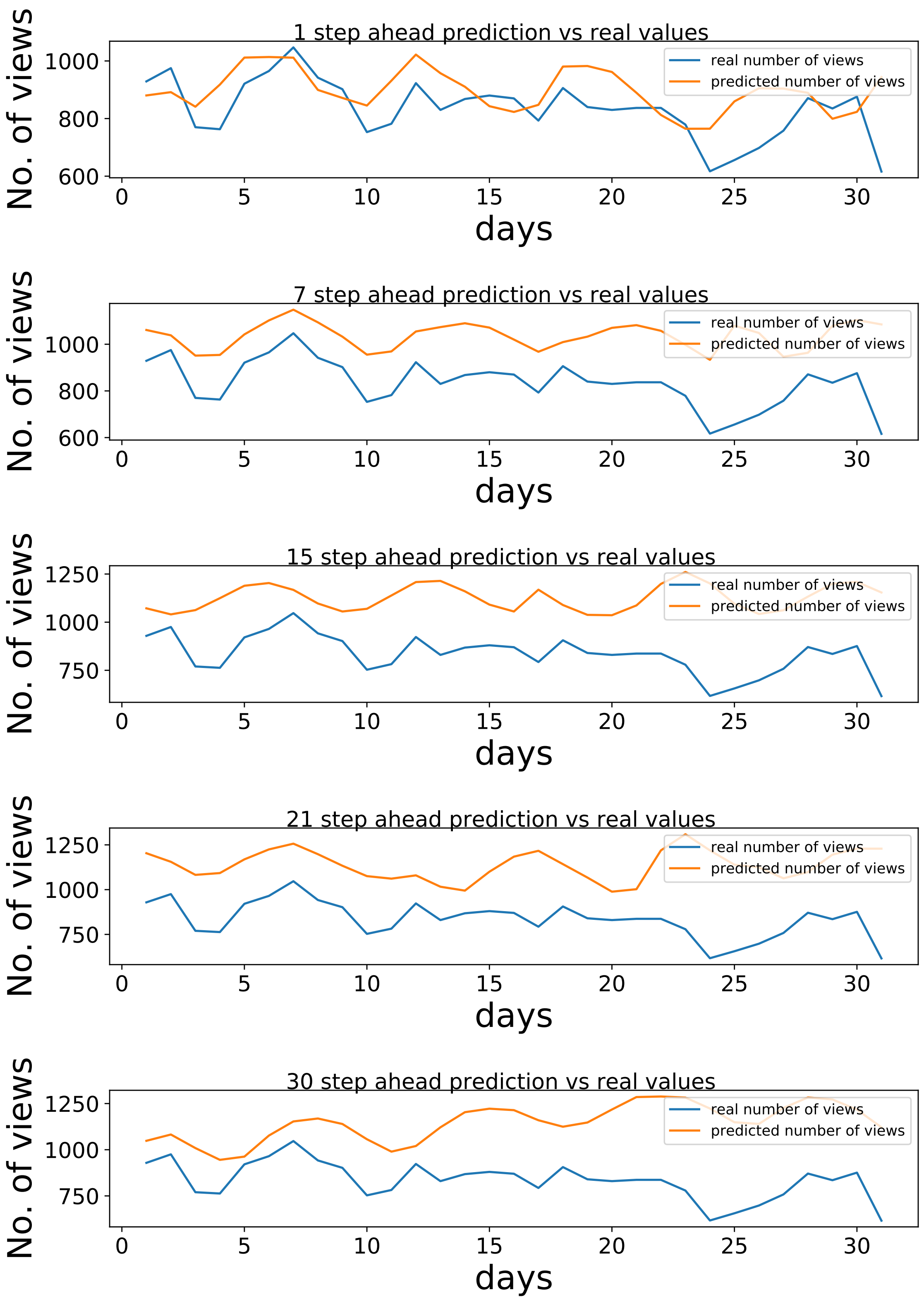


Figure 8: NRMSE for different prediction window ranges

Observation

During the process of tuning the parameters of the network it was observed that the data preprocessing affects the prediction by the network at higher extent than other factors. The reason for it, I believe, is that the preprocessing steps scales down the outlier in the training signal and also scales up the previously hidden features in the signal. Adding an additional sine wave to the input signal also significantly improves the prediction. This is because sine wave signal of period 7 gives the insight of weeks to the network. Therefore, on adding extra sine wave signal as the input, the network can easily identify this weekly pattern present in the main input signal. However, adding other additional inputs does not have a significant impact on the prediction. This could be because these additional are highly correlated to the main input signal and thus does not add much additional information to the network. During cross validation phase, with the smaller values of regularization coefficient, β it was observed that the predicted values in some of folds deviated highly from the real values and also from the mean values. This is because with very small regularization coefficient the output weights get bigger and thus even small perturbation in input signal lead to the strong perturbation in the predicted signal. The network gave the best result when the scaling factors for sf_W and sf_{Win} are in the range of 0.5 - 1.5 and 0.3 - 0.6 despite the fact that the difference in the performance is barely noticeable with any values assigned in that range.

5 Conclusion and Further Works

This section will present the conclusion of this project and will discuss possible further improvement in the project.
With this project, we tried to predict the number of views for Wikipedia pages, particularly the page titled "2002 FIFA World Cup". The main goal of this project was not to compete with the state of the art methods for web traffic forecasting but rather to try out the approach of using Echo State Network for this task. Although much work remains to be done and predictions are not good for longer prediction horizons, this study has shown that ESNs have great potential for this task and thus further improvement of the network setup and data preprocessing steps, the quality of prediction can be improved. This project is focussed on the prediction of web traffic for one particular page. Therefore, one of our future work would be to train the network to make prediction of web traffic for other Wikipedia pages and possibly multiple Wikipedia pages using single ESNs and even for pages from other websites. For this experiment, we have used the additional signals that are highly correlated to the main input signal. However, these highly correlated signals add very few information to the main input signal. Therefore in future, we would like to find the right signal to add to the input such that it is neither too less nor too highly correlated to the main input signal.

References

[1] Kanad Chakraborty et al. “Forecasting the behavior of multivariate time series using neural networks”. In: *Neural Networks* 5.6 (1992), pp. 961 –970. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(05)80092-9. URL: http://www.sciencedirect.com/science/article/pii/S0893608005800929.

[2] Paulo Cortez, Miguel Rocha, and Jose Neves. “Time Series Forecasting by Evolutionary Neural Networks”. In: *Idea Group Inc.* (2006). URL: http://www.irma-international.org/viewtitle/5363/.

[3] Jing Dai, Ganesh K. Venayagamoorthy, and Ronald G. Harley. “An Introduction to the Echo State Network and its Applications in Power System”. In: *: Intelligent System Applications to Power Systems, 2009. ISAP ’09. 15th International Conference* (2009). URL: http://ieeexplore.ieee.org/document/5352913/?arnumber=5352913.

[4] Georg Holzmann. *Echo State Networks in Audio Processing*. 2007.

[5] Georg Holzmann. “Echo State Networks in Audio Processing”. In: (Jan. 2007).

[6] Herbert Jaeger. “A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach”. In: 5 (Jan. 2002).

[7] Herbert Jaeger. “Echo state Network”. In: *Scholarpedia* 2.9 (2007). revision #183563, p. 2330. DOI: 10.4249/scholarpedia.2330.

[8] Herbert Jaeger. “Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication”. In: *Science* 304 (5667 2004), pp. 78–80. DOI: 10.1126/science.1091277.

[9] Herbert Jaeger. “Short Term Memory in Echo State Networks”. In: (Jan. 2002).

[10] Herbert Jaeger. *The "echo state" approach to analysing and training recurrent neural networks - with an Erratum note*. Tech. rep. Jacobs University Bremen, Sept. 2010.

[11] Gang Li et al. “Echo State Network with Bayesian Regularization for Forecasting Short-Term Power Production of Small Hydropower Plants”. In: *Energies* 8.10 (2015), 12228–12241. ISSN: 1996-1073. DOI: 10.3390/en81012228. URL: http://dx.doi.org/10.3390/en81012228.

[12] Da Liu, Jilong Wang, and Hui Wang. “Short-term wind speed forecasting based on spectral clustering and optimised echo state networks”. In: *Renewable Energy* 78.Supplement C (2015), pp. 599 –608. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2015.01.022. URL: http://www.sciencedirect.com/science/article/pii/S0960148115000294.

[13] Mantas Lukoševičius. “A Practical Guide to Applying Echo State Networks”. In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 659–686. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_36. URL: https://doi.org/10.1007/978-3-642-35289-8_36.

[14] Mantas Lukoševičius and Herbert Jaeger. “Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review 3, 127-149”. In: 3 (Aug. 2009), pp. 127–149.

[15] Spyros G. Makridakis, Steven C. Wheelwright, and Rob J. Hyndman. *Forecasting: Methods and Applications, 3rd Edition*. Wiley, 1998.

[16] Rojaa Ramani Matlakunta. “Web Traffic Prediction for Online Advertising”. Master Thesis. Auckland University of Technology, 2011, pp. 63–64.

[17] Sibarama Panigrahi, Yasobanta Karali, and H. S. Behera. “Time Series Forecasting using Evolutionary Neural Network”. In: *International Journal of Computer Applications(0975-8887)* 75.10 (Aug. 2013), pp. 13–17.

[18] Ernesto Rodriguez. “Training Echo State Networks with Short Segments of Motion Capture Data”. Bachelor’s Thesis. Jacobs University, 2013, pp. 8–9.

[19] Otto Seiskari. *8th place with Kalman filters*. https://www.kaggle.com/c/web-traffic-time-series-forecastsing/discussion/43727. [Online; accessed 12-December-2017]. 2017. URL: https://www.kaggle.com/c/web-traffic-time-series-forecastsing/discussion/43727.

[20] Anne Senter. *Time Series Analysis*. http://userwww.sfsu.edu/efc/classes/biol710/timeseries/timeseries1.htm. Blog. 2011. (Visited on 12/05/2017).

[21] Artur Suilin. *kaggle-web-traffic*. https://github.com/Arturus/kaggle-web-traffic. 2017.