JACOBS
UNIVERSITY

Web Traffic Prediction using Echo State Networks

by

Lalit Singh

Bachelor Thesis in Computer Science

Prof. Herbert Jaeger
Bachelor Thesis Supervisor

Date of Submission: May 6, 2018

Jacobs University — Focus Area Mobility

With my signature, I certify that this thesis has been written by me using only the indicates resources and materials. Where I have presented data and results, the data and results are complete, genuine, and have been obtained by me unless otherwise acknowledged; where my results derive from computer programs, these computer programs have been written by me unless otherwise acknowledged. I further confirm that this thesis has not been submitted, either in part or as a whole, for any other academic degree at this or another institution.

Signature Place, Date

Abstract

Consider this a separate document, although it is submitted together with the rest. The abstract aims at another audience than the rest of the proposal. It is directed at the final decision maker or generalist, who typically is not an expert at all in your field, but more a manager kind of person. Thus, don't go into any technical description in the abstract, but use it to motivate the work and to highlight the importance of your project.

(target size: 15-20 lines)

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 organization forward. Time series forecasting is an important tool under this scenario where the goal is to predict the behavior of complex systems solely by looking at patterns in past data. A time series is a collection of periodic ordered observations that appears in multiple range of domains such as literature, agriculture, finance, media, etc., just to name a few [1]. Time series are very complex because each observation is dependent upon the previous observations and often is influenced by more than one previous observations [2]. Hence, this research mainly focuses on 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 predicted web traffic because they need to distribute advertisements to the selected websites based on the predicted web traffic [3]. 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 has been performed using various statistical-based methods [4]. The major drawback of most of the 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 suffered from nonlinearity of underlying data generating process [5]. In [3], 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 prediction accuracy [3]. Work in [6] 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 [7], 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 organized 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 [8].

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) [9]. 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 Prof. Jaeger in 2001 and are regarded as the closest representation of the learning process of the human brain [10]. ESNs can effectively solve all 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 [11, 9]. Therefore, we have chosen to use ESNs for the prediction of web traffic.

1.3 Objective

Our main objectives to conduct this research are listed below:

- To evaluate the performance of ESN for the prediction of web traffic.
- To optimize the parameters of ESN for it's best performance.

2 Statement and Motivation of Research

This part should make clear which question, exactly, you are pursuing, and why your project is relevant/interesting. This is the place to cite relevant literature. Where does your project extend the state of the art? What weaknesses in known approaches do you hope to overcome? If you have carried out preliminary experiments, describe them here.

(target size: 3-5 pages)

3 Background Concepts

3.1 Echo State Networks

3.2 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 [12]. 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].

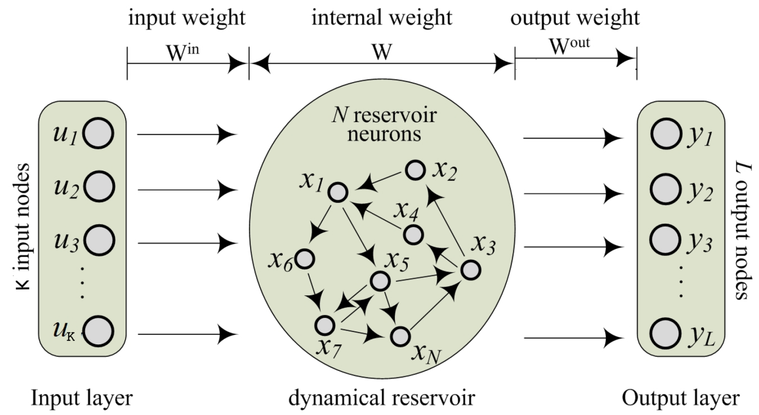


Figure 1: The basic schema of Echo State Network. Image source[14].

3.3 Formalism

3.3.1 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) [**?**].

3.3.2 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 $\mathbf{u}(n) = (u_1(n), \ldots, u_K(n))$, of internal units are $\mathbf{x}(n) = (x_1(n), \ldots, x_N(n))$ and of output units are $\mathbf{y}(b) = (y_1(n), \ldots, y_L(n))$. Real valued connection wights 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 in the network are collected in matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$, and for the output weights are collected in matrix $\mathbf{W}^{out} \in \mathbb{R}^{L \times (K+N+L)}$ [15]. 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" [**?**].

The activation of internal units are updated according to the discrete time state update equation given below:

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

where $\mathbf{u}(n)$ is the externally given input,$f()$ denotes the 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, $i = 1, \ldots, n_{max}$, 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 by the equation below:

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

$\mathbf{g} = (g_1, \ldots, g_L)$ is unit output activation function (in our case identity).[12].

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[12].

4 Description of the Investigation

This is the technical core of the thesis. Here you lay out your how you answered your research question, you specify your design of experiments or simulations, point out difficulties that you encountered, etc.

(target size: 3-4 pages)

4.1 Data Description and Preprocessing

The data set 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 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 or 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 a particular Wikipedia page received on that date. The total length of each time series is 550. There are some missing values 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, we have chosen the traffic generated for 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 following steps are followed:

4.1.1 Step 1

The time series data used in this experiment has very large values (≈ 7000) as seen in the the Figure 2. Therefore, the time-series data is squeezed component wise 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

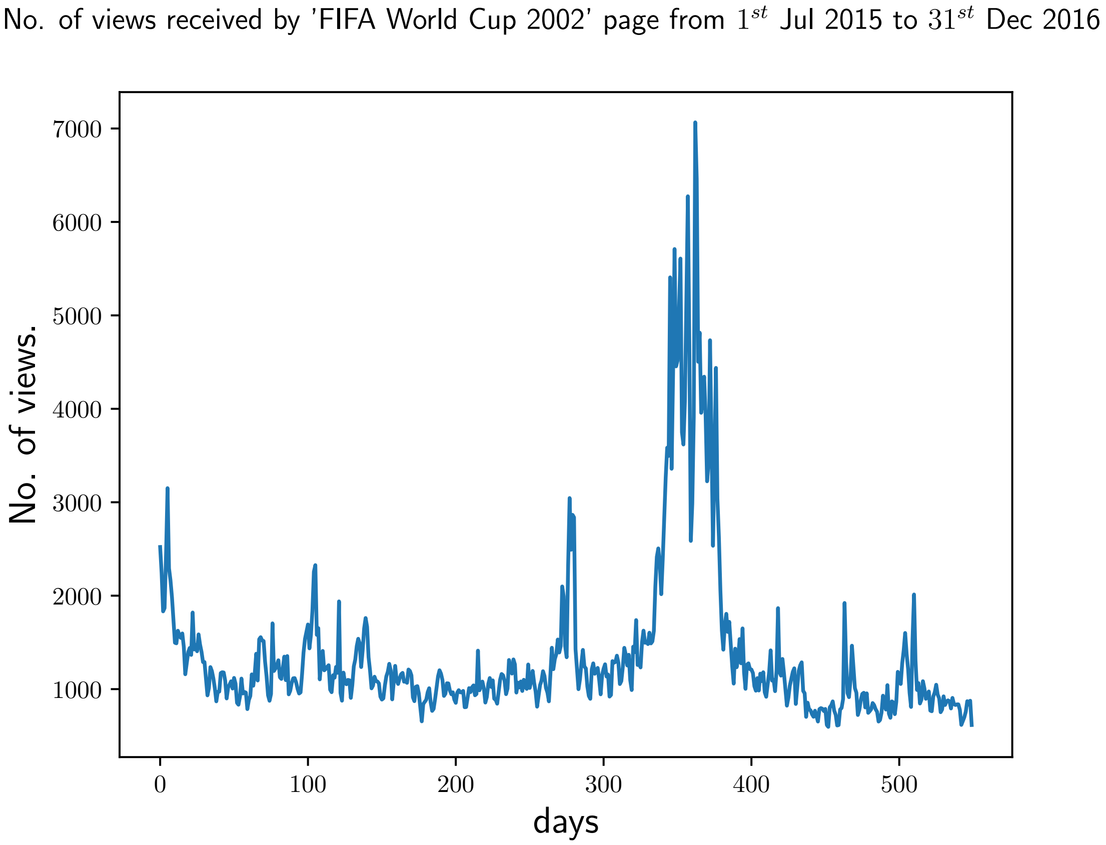
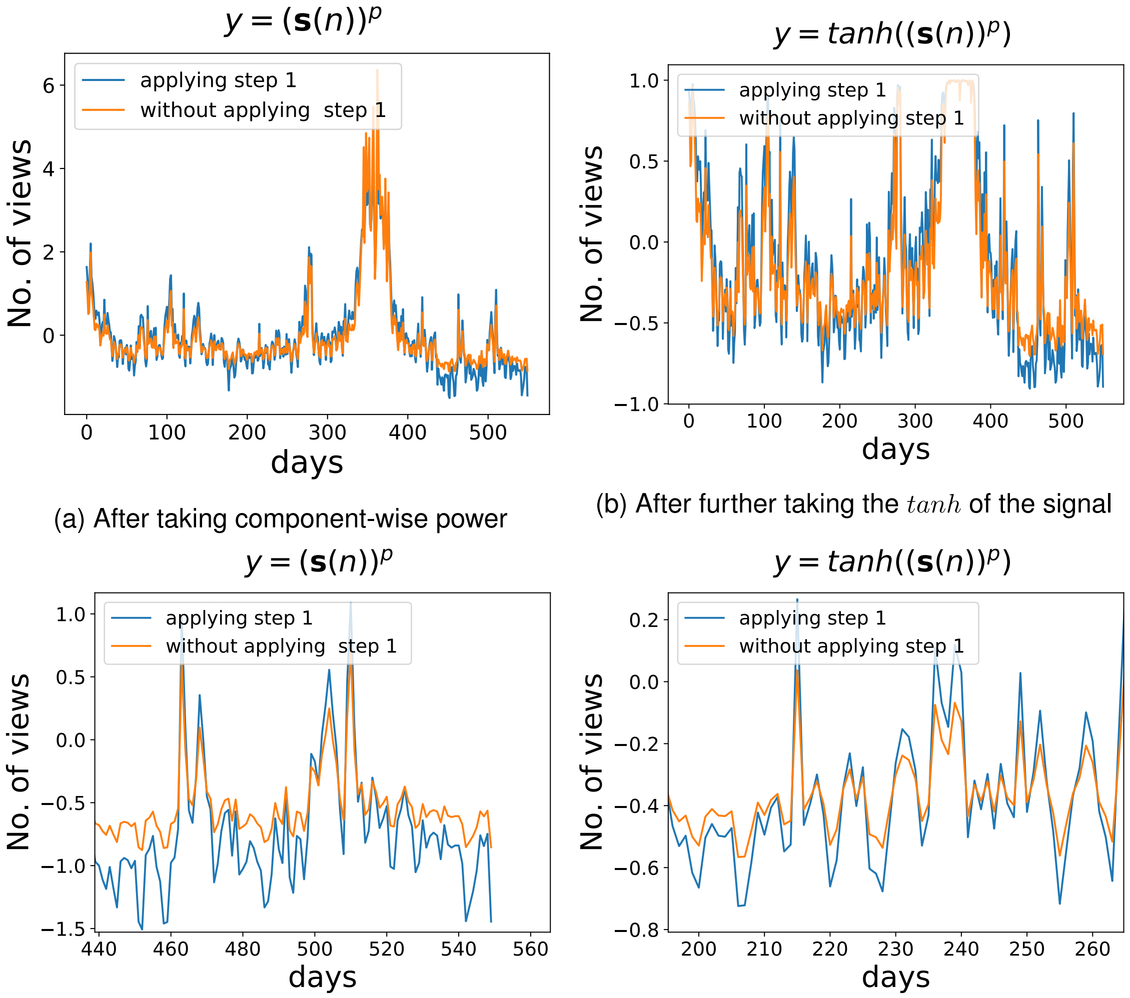


Figure 2: Raw data which has number of views of Wikipedia Pages on each date from 1st of July 2015 to 31st of December 2016.

$\mathrm{s}(n) = (s(1), \ldots, s(n))$. The power p that need to be raised to each component of time-series is calculated using Eq. 4.

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



(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

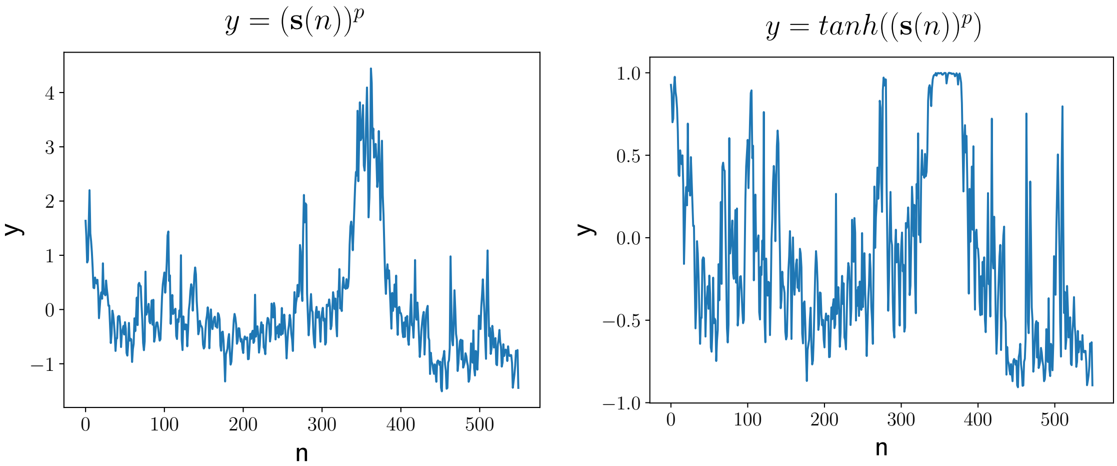
Figure 3: Preprocessing steps

4.1.2 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 component wise by its standard deviation to make its variance unit.

4.1.3 Step 3

The resultant time series from step 2 is further applied a $tanh$ function component wise. 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 values in signal is penalized 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 (zoomed in version in Fig. 3 c) the yellow signal generated with the application of only step 2 has lesser wiggliness 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 wiggliness 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 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 wiggleness and scales down the higher peaks such that the values at any time point in signal is within the range of -1 to 1.

4.2 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 that on related subject matters might receives 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 following approach to find suitable additional input for the network.

1. First of all a sine wave of period 7 was searched which has highest correlation with the main input signal. 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 with has 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 highest correlation to the pages given in Table 1.

Table 1: Information about the additional signal

Wikipedia page Title	Wikipedia version	correlation coefficient
Fuball Weltmeisterschaft 1994	de.wikipedia.org	0.92
2010 FIFA World Cup	en.wikipedia.org	0.94
Copa Mundial de Ftbol de 2010	es.wikipedia.org	0.91
Fuball-Europameisterschaft 2008	de.wikipedia.org	0.90
Fuball-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
Fuball-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
Fuball-Europameisterschaft 1988	de.wikipedia.org	0.92
2006 FIFA World Cup	en.wikipedia.org	0.96
2002	ru.wikipedia.org	0.90
Fuball-Weltmeisterschaft 2002	de.wikipedia.org	0.93
2002 FIFA World Cup	en.wikipedia.org	0.96
Fuball-Weltmeisterschaft 2014	de.wikipedia.org	0.91
2014 FIFA World Cup	en.wikipedia.org	0.91
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 these additional signals were a scaling factor sf_{add}/NUM, where NUM is the total number of such additional inputs. To determine the correlation between two time series we have used Pearson’s correlation coefficient.

4.2.1 Pearson correlation coefficient

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

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

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

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 $\bar{y}$

So to give the insight of weeks and months, two additional input signals with periods 7 and 30 $\pm$ 2 were introduced to the network. These periodic signals were preprocessed (applied log10 and shifted by its mean) before feeding it to the network.

4.3 Network Setup

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 build out of it’s own ”echos” in the Dynamic Reservoir. [**?**]
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. Then $\mathbf{W}$ matrix is normalized by its maximum eigenvalue. There are $K = 22$ input neurons that receive each of the 21 input signals and $L = 31$ output neurons each of which makes $t = 1, \ldots, 31$ step ahead prediction.

In order to achive 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 [**?**]. In this experiment, the network weight matrix, $\mathbf{W}$ has different weight regions to create different capability of 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 below:

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

Values in the weight matrix for in all the regions are drawn from the uniform distribution of range -1 to 1 and 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) [12]. The weight matrix $\mathbf{W}_0$ is normalized to $\mathbf{W}_1$ with it's 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 $sf_{W}in$ and sf_B respectively.

4.4 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 one indicates that the compared two signal y and r are completely unrelated to each other. NRMSE of value close to zero indicates that the model might have overfitted and could give bad performance on testing data set.

4.5 Regularization

In order to access the quality of the prediction produced by the training of ESN, we regularly monitor the actual obtained 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) .

4.6 Leaking Rate

The leaking rate $\alpha \in \mathbb{R}^{N\times 1}$ of the reservoir nodes can be regarded as the speed of the reservoir update dynamics discretized in time. So 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 the one dimensional vector can be optimized for the better performance of the network however this increases the number of parameters that needs to be optimized linearly with 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 needs to be optimized.

4.7 Cross Validation

In length of available timeseries data is 550 out of which we leave out the last 31 data points for the testing purpose once 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 removing a part of it for the validation part 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 increased 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 first choice because it exactly satisfies previously stated requirement.

4.7.1 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 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**$_k$, is average fo the error in each fold:

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

4.7.2 Computational Efficiency in K-fold cross validation

As mentioned in section 4.7.1, 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 which can be computationally 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 $i = 1, \dots, n_{max}$ (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 save to assume that the effects of the arbitrary initial state have washed away and the network gives a pure reflection of the input signal [**?**].

After removing initial washout from the state collection matrix $\mathbf{S}$, it is divided in K folds in horizontal axis. In each iteration of loop that runs from $i = 1, \dots, K$ 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 way we reduce the computation cost of training the network during cross validation phase.

4.8 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 [**?**].

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

4.9 Teacher signals

5 Evaluation of the Investigation

This section discusses criteria that are used to evaluate the research results. Make sure your results can be used to published research results, i.e., to the already known state-of-the-art.

6 Results

After the optimization of the variables during cross validation phase of the experiment the network is retrained using all the available signal. 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 $\mathbf{W}^{out}$ 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 us used to make the prediction for the number of views for the month of December 2016. The accuracy of each of prediction range is calculated as NRMSE. These error measures are plotted in the Figure 5. The predicted number of views and the actual number of views for different prediction horizon are plotted in figure 6. As seen in Figure 5, usually the NRMSE for the smaller prediction horizon is smaller. For longer prediction horizon the NRMSE is bigger. The NRMSE values doesn't increases linearly with the prediction horizon. One reason for this could be, I assume, that some of the features like biweekly repetition of triweekly correlation of the data might be strong that any other frequency.
The optimizable parameters that worked best for this experiment are presented in the table 2.

Table 2: Optimized values for control parameters

Parameters		Optimized value
$sf^{\mathbf{W}}$		1.0
$sf^{\mathbf{Win}}$		1.9
$sf^{\mathbf{B}}$		1.4
α	HIGH	5.05
sf_{sine}		
sf_{add}		
α	FAST	
	MEDIUM	
	SLOW	

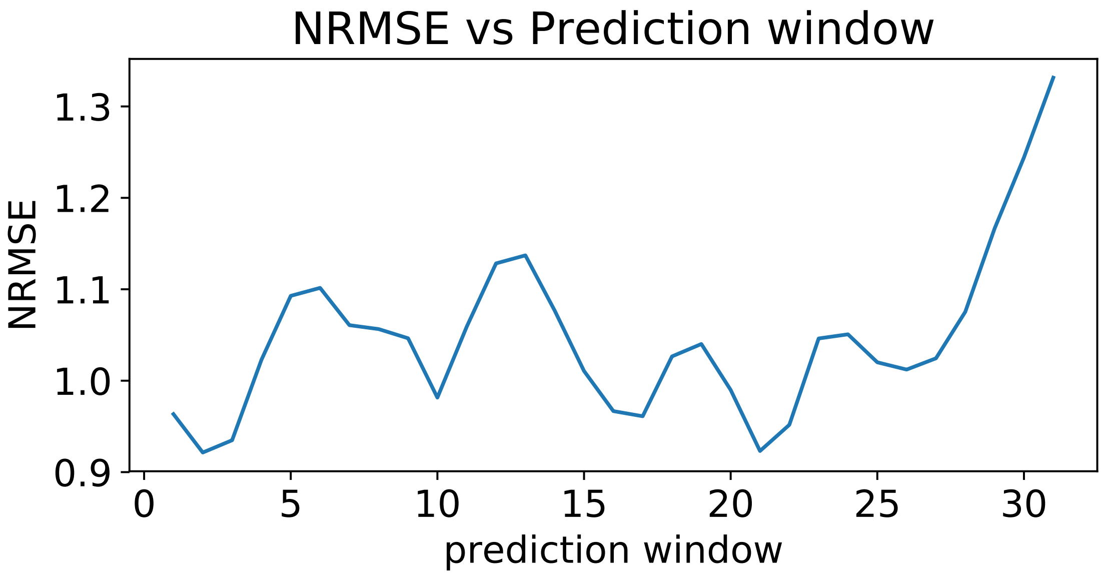


Figure 5: NRMSE for different prediction window ranges

6.1 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 beleive, is that the preprocessing steps followed 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 signal that represents the view received by the wikipedia page as a weekly trend. Therefore on adding extra sine wave signal as the input, the network can easily identify this weekly pattern. However additional inputs, number of views for the related Wikipedia pages does not have signifincant impact on the prediction. This could be because these additional are highly correlated to the main input signal and thus does not provide much additional information to the network about the signal. 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 coefficent the output weights gets bigger thus even small peturbation in input signal lead to the strong perturbation in the prediction. The network gave best result when the scaling factors for sf_W and $sf_W in$ are in the range of 0.5 - 1.5 and 0.3 - 0.6 despite the fact that the difference in the performace is barely noticable with any values assigned in that range.

7 Conclusions

Summarize the main aspects and results of the research project. Provide an answer to the research questions stated earlier.

(target size: 1/2 page)

This section will present conclusion of this project and will discuss possible further im-

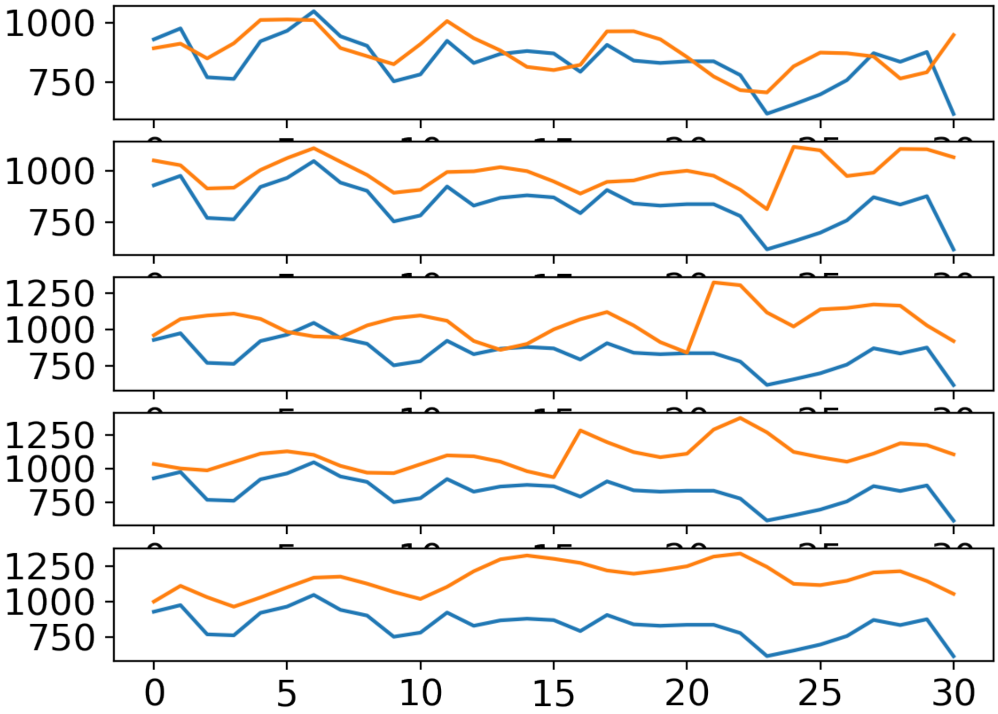


Figure 6: NRMSE for different prediction window ranges

provement 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”.

References

[1] Paulo Cortez, Miguel Rocha, and Jose Neves. Time series forecasting by evolutionary neural networks. *Idea Group Inc.*, 2006.

[2] Anne Senter. Time series analysis. http://userwww.sfsu.edu/efc/classes/biol710/timeseries/timeseries1.htm, 2011.

[3] Rojaa Ramani Matlakunta. Web traffic prediction for online advertising. Master thesis, Auckland University of Technology, March 2011.

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

[5] Sibarama Panigrahi, Yasobanta Karali, and H. S. Behera. Time series forecasting using evolutionary neural network, 2013.

[6] Kanad Chakraborty, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. Forecasting the behavior of multivariate time series using neural networks. *Neural Networks*, 5(6):961 – 970, 1992.

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

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

[9] Da Liu, Jilong Wang, and Hui Wang. Short-term wind speed forecasting based on spectral clustering and optimised echo state networks. *Renewable Energy*, 78(Supplement C):599 – 608, 2015.

[10] Herbert Jaeger. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80, 2004.

[11] Georg Holzmann. Echo state networks in audio processing, 2007.

[12] Herbert Jaeger. Echo state network. *Scholarpedia*, 2(9):2330, 2007. revision #183563.

[13] Mantas Lukoševičius. *A Practical Guide to Applying Echo State Networks*, pages 659–686. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[14] Gang Li, Bao-Jian Li, Xu-Guang Yu, and Chun-Tian Cheng. Echo state network with bayesian regularization for forecasting short-term power production of small hydropower plants. *Energies*, 8(10):1222812241, Oct 2015.

[15] Herbert Jaeger. The ”echo state” approach to analysing and training recurrent neural networks - with an Erratum note. Technical report, Jacobs University Bremen, September 2010.

[16] J. Sample. On the Possibility of the Impossible. Technical report, Jacobs University Bremen, September 2010.