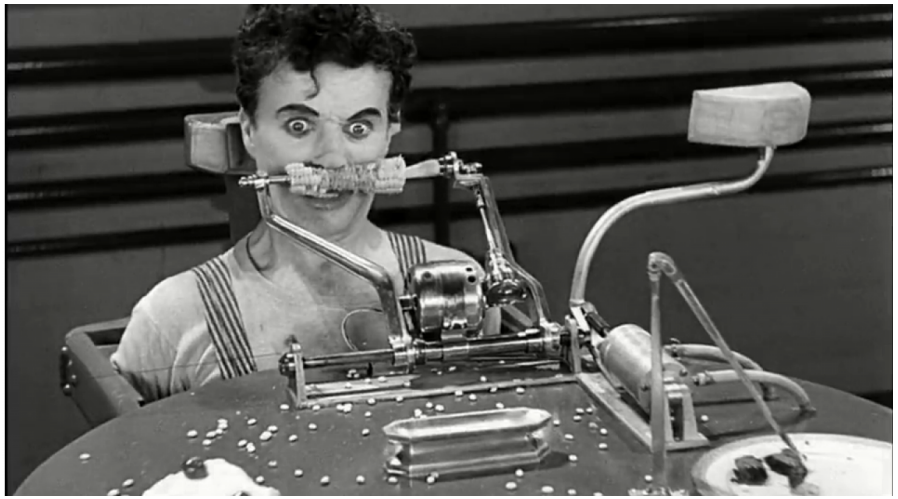
SHORT TERM TIME SERIES PREDICTION: THE RESERVOIR COMPUTING APPROACH

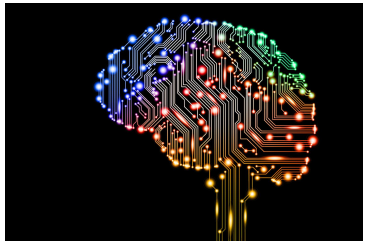
Alper Alimoglu

Background

- Mankind has always looked for ways design machines to automate certain tasks to ease their life and require less human intervention. Today electronic devices have become an important part of our everyday lives and standard computer programs are quickly processing large amounts of predictable data in a deterministic way.



Background



- Humans solve a widespread class of problems on a daily basis. Tasks that requires **intelligence** (means to *understand* in Latin) such as facial recognition, language processing, writing novels, playing instruments, and playing sports are skilled by most people yet no device can reach to same unique level of intelligence, thinking and solving.
- When the user executes a certain task multiple times, the computer program will not understand its pattern. The implement for computer program which autonomously learns the pattern of a executed task and concerned with making computers behave like humans is what powers research in artificial intelligence (AI) and machine learning (ML).

What Is Machine Learning?

- Machine learning is a research field that tries to perform algorithms by allowing a computer to autonomously learn a model from many of example data or past experiences.
  'What's going to happen in the future?' Past experiences hold the answers, and ML is the leading edge method for interpreting and using the data.

"A computer program is said to learn from experience E with respect to some task T and some performance measure P(probability of winning), if its performance on T, as measured by P, improves with experience E." (Mitchell, 1997).

"Field of study that gives computers the ability to learn without being explicitly programmed" (Arthur Sanuel 1959).

What is Time Series?

- A time series is represented by a sequence of continuous numbers, which is measured in a time spaced at equally discrete intervals (Δt).

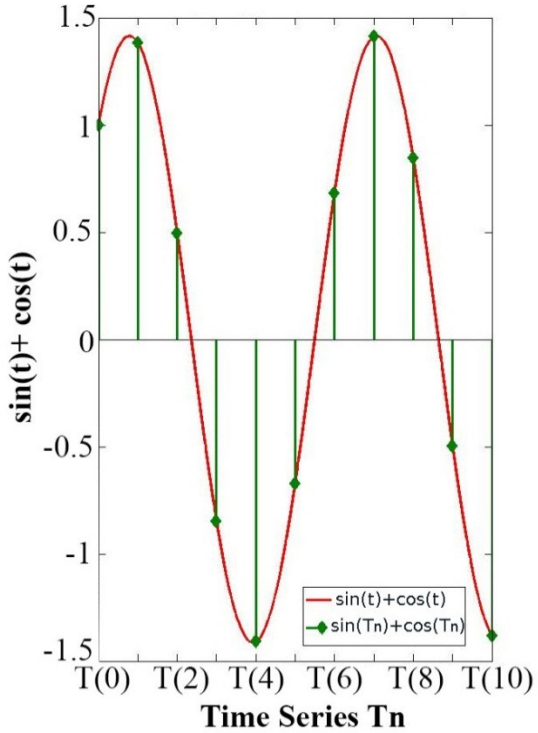


Figure 1: Plot of function f(t)=cos(t)+sin(t) with timer series plot overlay.

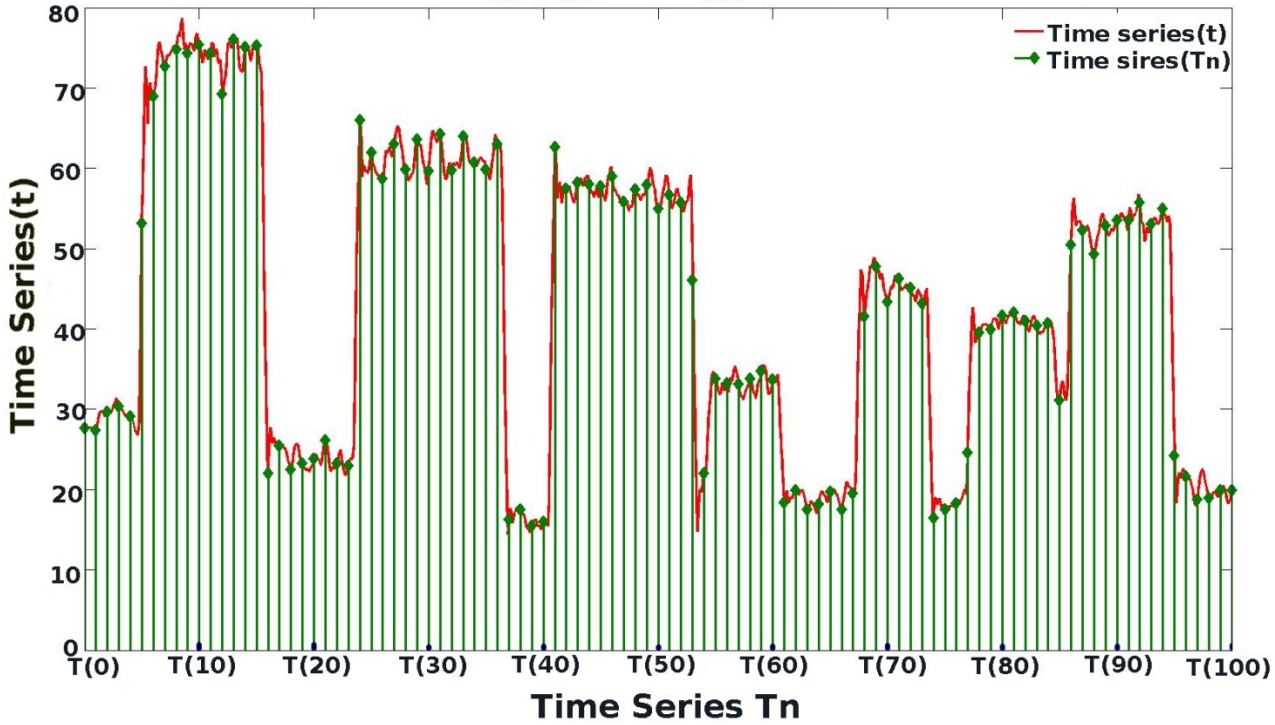


Figure 2: Time series(t) with time series plot overlay.

Example of Time Series

- Stock market (closing value each day)
- Sunspot activity (each day)
- Electricity demand for a city
- Number of births in a community
- Air temperature in a building
- Workload of a computer-server

These phenomena may be discrete or continuous.

Sometimes data have to be aggregated to get meaningful values.

Ex: births per minute might not be as useful as births per month.

Possible Types of Processing

- Predict future values of x[t]
- Classify a series into one of a few classes:
  - “price will go up”
  - “price will go down” —sell now
  - “no change”
  - Describe a series using a few parameter values of some model.
  - Transform one time series into another
    - oil prices => interest rates

Classify Series

- The prediction accuracy required by the decision problem will have an effect on the model generated from the prediction system. It will serve as the foundation for the main prediction accuracy for each unique task. An important characteristic of a accurate prediction system is its ability to obtain optimal performance in the face of uncertainty in the future.

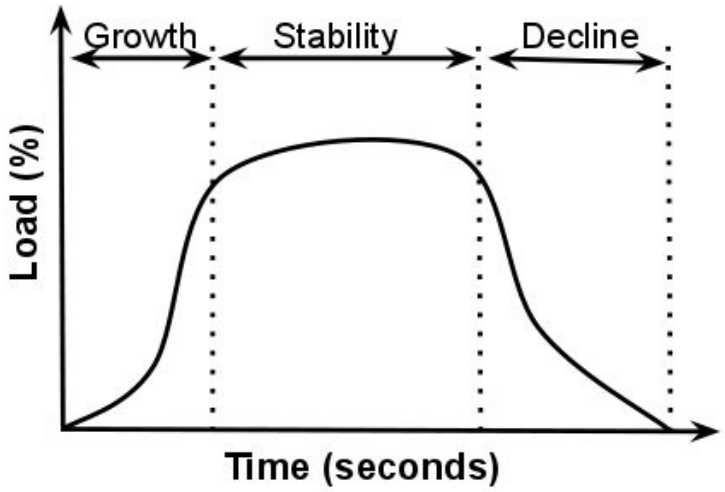
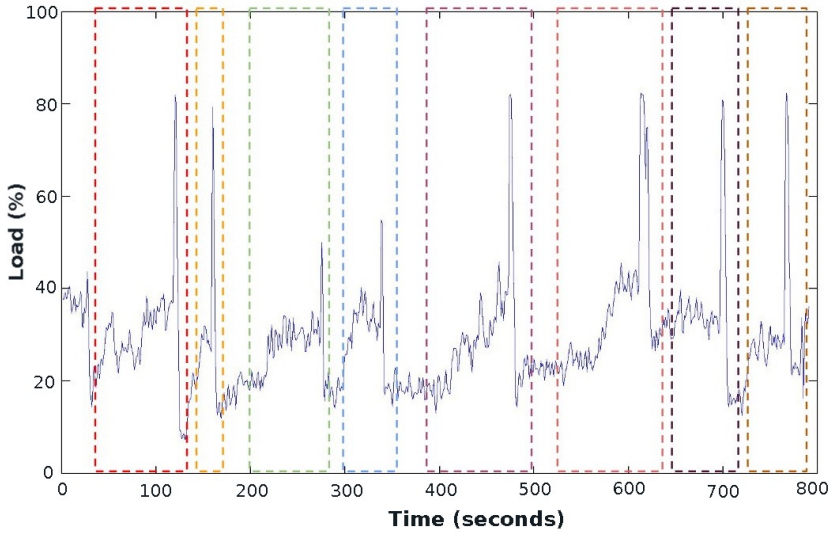


Figure 3: Example of a spike's life cycle.

In certain time series predictions, the prediction of significant changes lead to a increase or decrease in the variables value and has a higher priority than predicting the variables' exact value. In this case, a switch will cause the process to become out-of-control during the predicted time.

Data Centres Workload Time Series Example

- Most people and research centers are interested in predicting the future. This is the main problem of time series prediction. To predict time series, it is necessary to express the behavior of the signal by a mathematical model that can be applied in the future.
- The main focus of this work is to summarize the approach to make time series prediction more accurate in relation to Reservoir Computing by learning from previous experience.



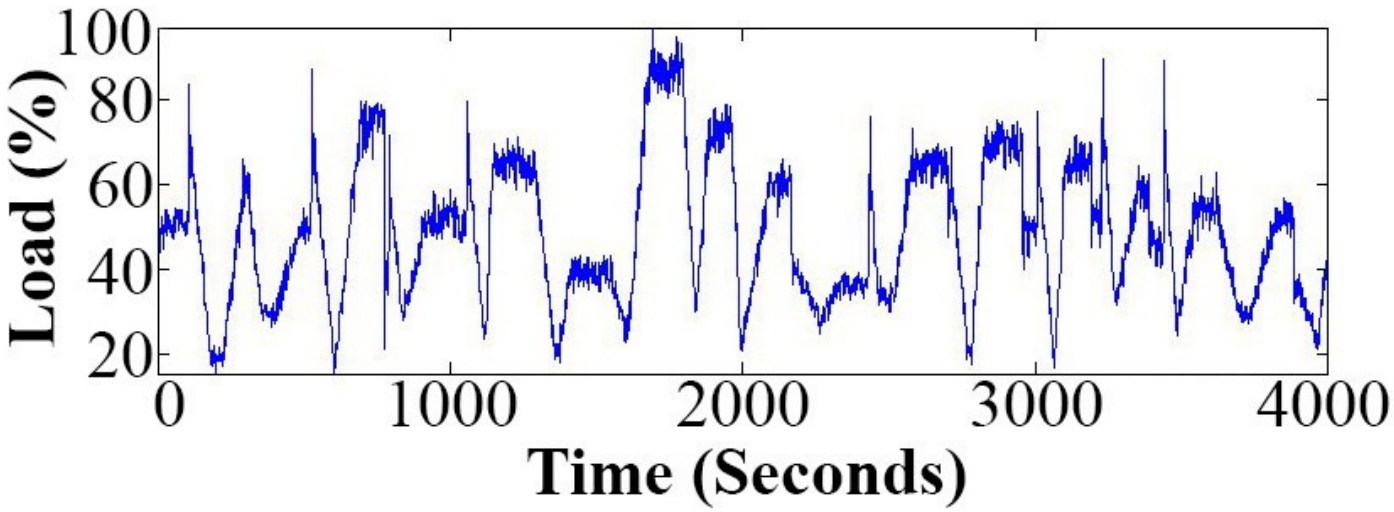


Figure 4: Real life example of a plotted workload sequence: the workload usage by the data centre.

Steps of Predicting the Future

Extending backward from time *t*, we have time series {x[t], x[t-1], x[t-2], …}. From this, we now want to estimate *x* at some future time:

$$x[\, t+s \,] = f(\, x[t], x[t-1], x[t-2], \ldots \,)$$

s is called the *horizon of prediction*. Let's predict one time sample into the future, => $s = 1$

The steps we have to follow:

1. Assume a generative model.
2. For every point x[*ti*] in the past, train the generative model with what preceded ti as the *Inputs* and what followed *ti* as the *Desired*.
3. Now run the model to predict x[t+s] from {x[t], x[t-1], x[t-2], …}.

Embedding

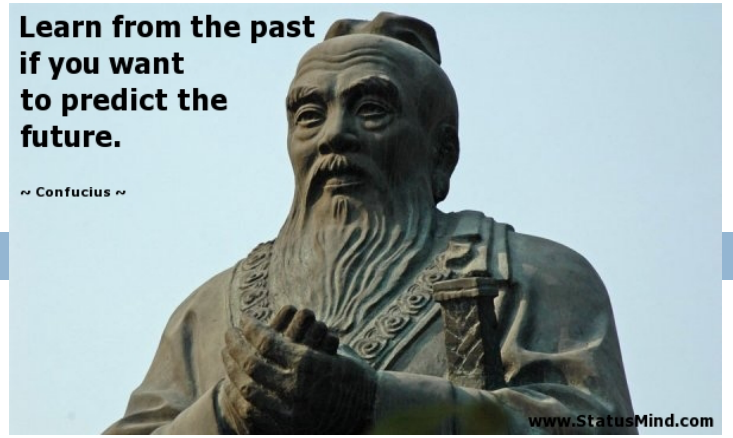
Time is constantly moving forward (like a flowing river). Temporal data is hard to deal with.

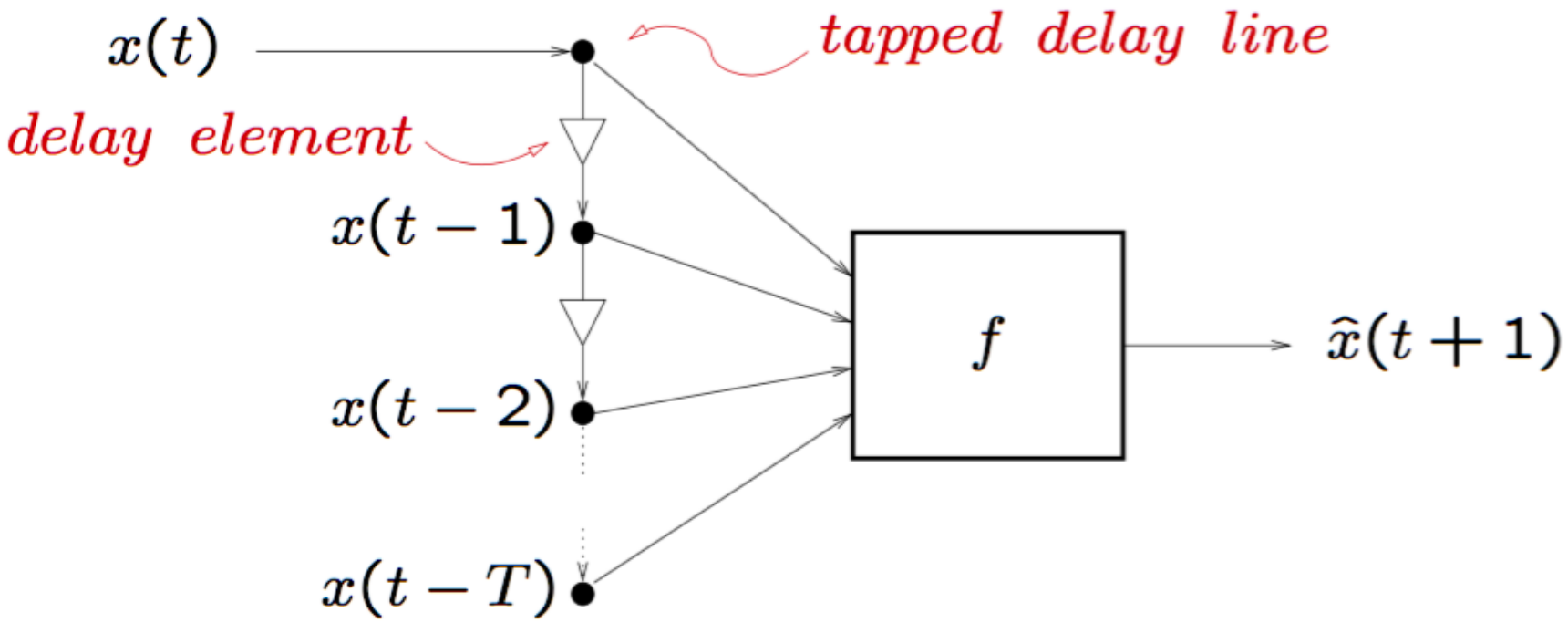
If we set up a shift register of delays, we can retain successive values of our time series. Then we can treat each past value as an additional *spatial* dimension in the input space to or predictor.

This implicit transformation of a one-dimensional time vector into an infinite-dimensional spatial vector is called *embedding*.

The input space to our predictor must be finite. At each instant t, truncate the history to only the previous *d* samples. d is called **the embedding dimension**.

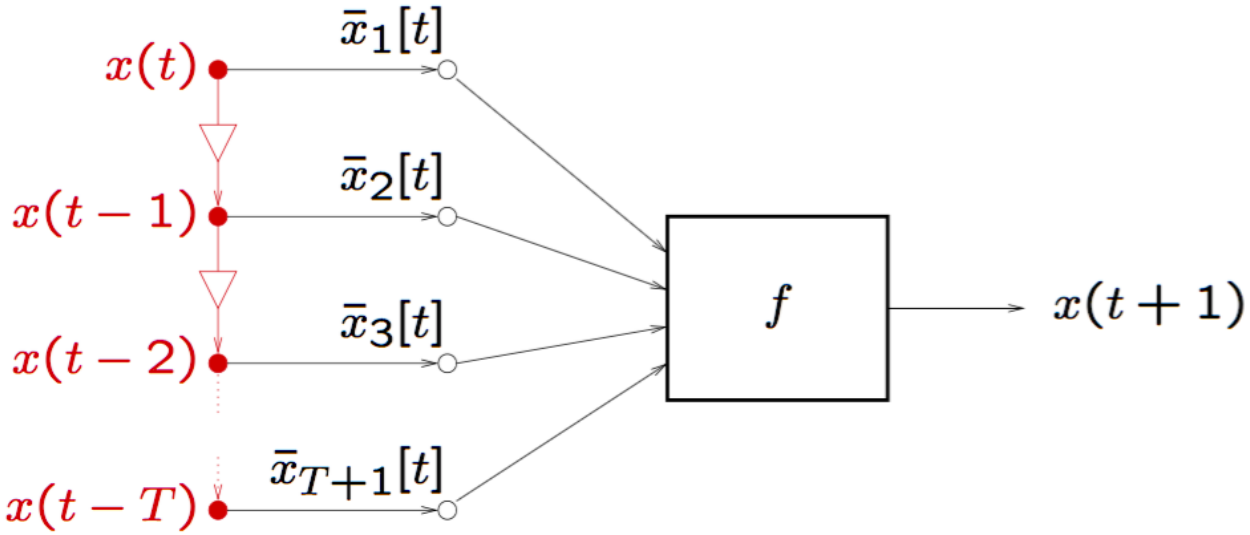
Using the Past to Predict the Future





Memory Models

Uniform sampling is simple but has limitation. **Depth** is how far back memory goes.



Can only look back T equal distant time steps. To look far into the past, T must be large.

Large T => complicated model: many parameters, slow train.

Horizon of Prediction

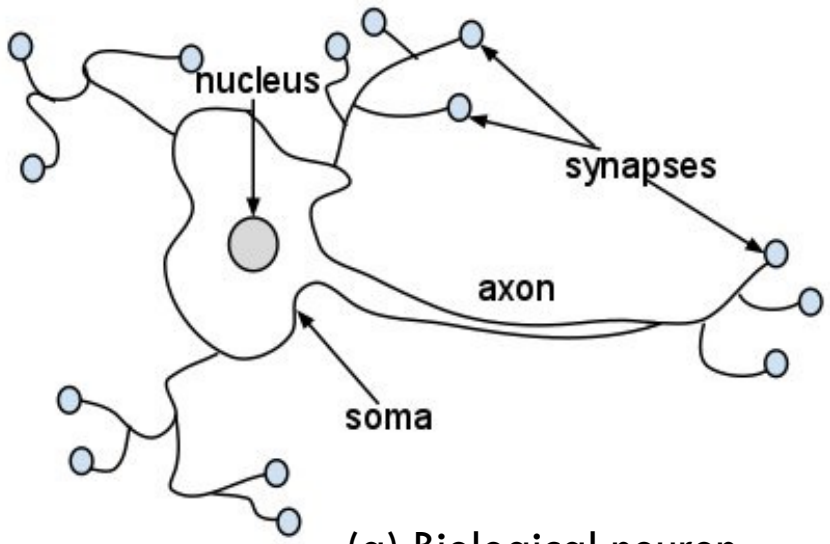
We have talked about predicting the next sample in a time series. What if we need a longer forecast, ex: not x[t+1] but x[t+*s*], with the horizon of prediction s > 1?

Three options:

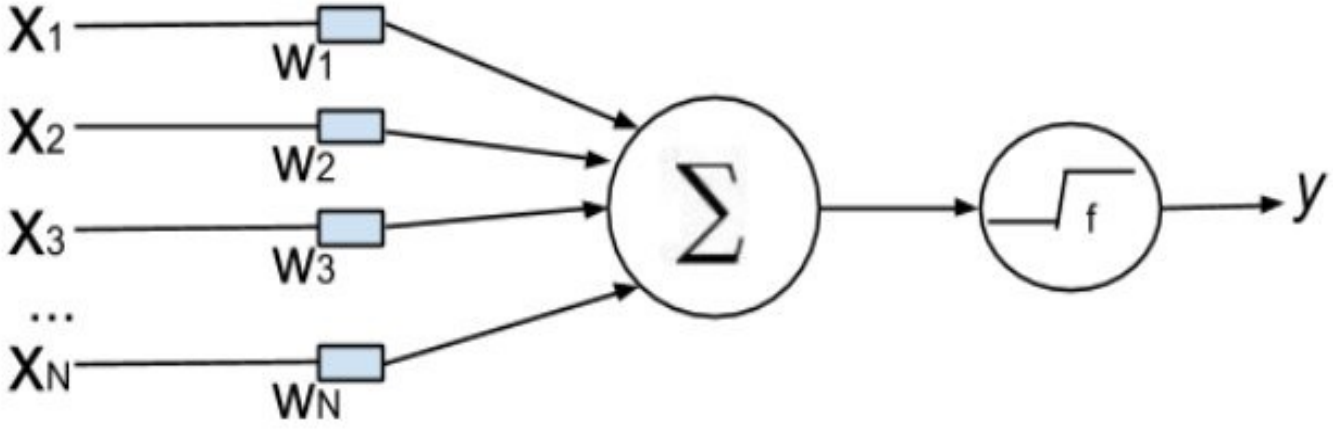
- Train on {x[t], x[t-1], x[t-2], … } to predict x[t+s]. (direct prediction)
- Train to predict all x[t+i], $1 \geq i \geq s$ (efficient for small s)
- Train to predict x[t+1] only, but iterate to get x[t+*s*] for any *s*.

Artificial Neural Networks

- Artificial Neural Networks (ANNs), as the term implies, is a replica of the biological neural networks found in the human brain. Biological neurons and artificial neurons represent a structural similarity in regards to functionality in the brain.
- Each neurons is a simple processor; it is composed of a basic data path that executes primitive operations. The neuron adds all weighted inputs received and outputs the threshold of current input values. Each linear threshold function, (Mc-Culloch–Pitts neuron), assemble together to perform universal computation for the chosen weights.



(a) Biological neuron



(b) Artificial neuron

Artificial Neural Networks

- Neural networks such as multi-layered are complex to train, tune, and are difficult to scale up completely.
- Each neurons receives weighted values from the incoming neurons to which it is connected, sums these values, computes its own output by applying some function and transmits these output values via the outgoing weight connections to the other nodes.

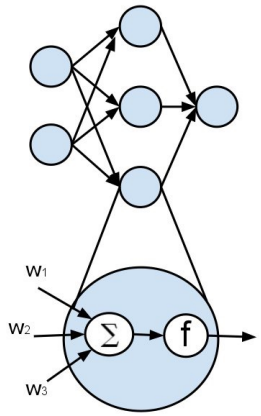


Figure *: A schematic representation of a simple network of neurons. Internal operation of a neuron: The total weight amount of inputs is passed through a nonlinearity.

$$y_k[n] = f\left(\sum_{i=1}^{M} w_{ik}\, y_i[n-1]\right),$$

The input-output behavior of an artificial neuron with index k at time-step n represented:

Feed-Forward

- **Feed-Forward:** During the training time the artificial network understands patterns between input and output, thus creating the learning process. The nodes of the network are divided into layers, with information flowing only to consecutive and not backwards. Past elements that are covered by a limited time window, Figure 5, are fed into the network to predict future time steps. The problem is that the number of inputs presented to the network rapidly becomes large and temporal ordering is eliminated.

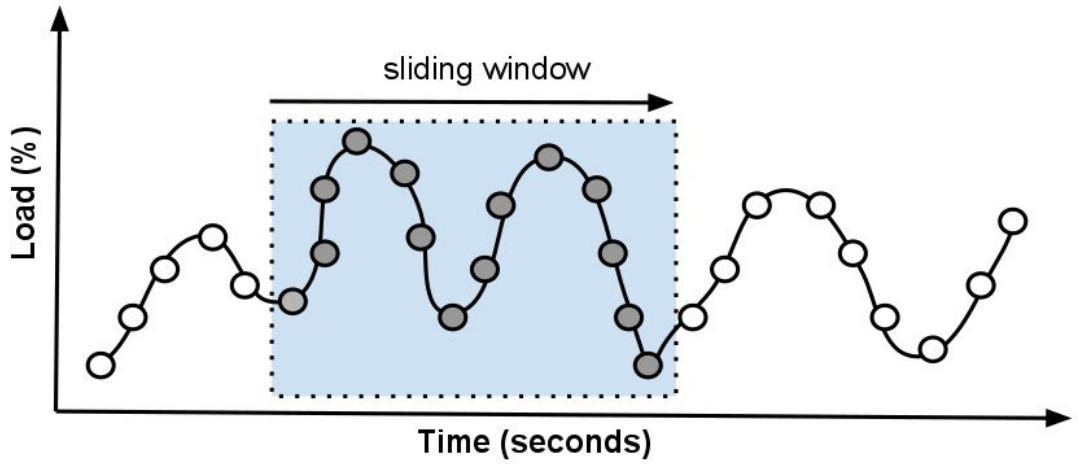


Figure 5: A schematic representation of a sliding window. Past elements which are covered by a limited time window(
white nodes) are mapped onto a future time step
(gray nodes). After that, the window is moved one time step at a time in future.

Problems with Feed-forward learning

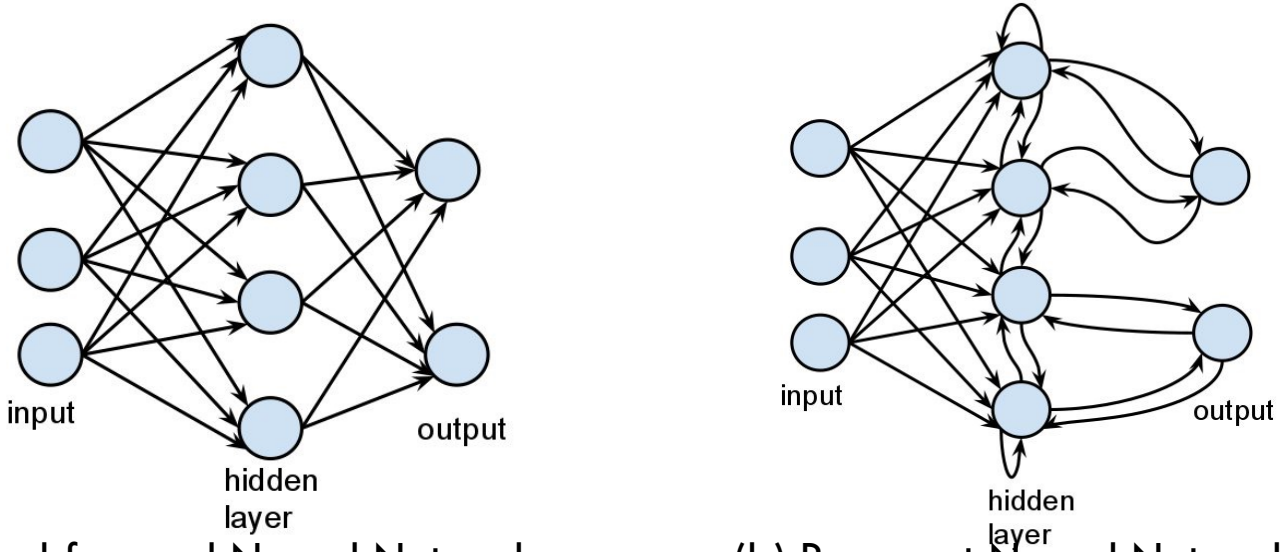
- One of the main problems with feed-forward NNs is that they are not designed to model temporal data like time series.
- A frequently used technique to deal with temporal data are to use a time window. Past elements that are covered by a limited time window, Figure 5, are fed into the network to predict future time steps. The problem is that the number of inputs presented to the network rapidly becomes large and temporal ordering is eliminated. In the example Figure 5, **future load values depend on what happened more recently than what happened further in the past.**
- On the other hand, future load values might also influenced by cyclic effects such as weather cycles.
- A more natural way to solve this is by a technique that is time-aware. This is made possible by allowing recurrent connections in the hidden layer, see Figure 6b, which is known as RNN.
- Convergence is slow, computationally expensive, and can lead to a poor local minima.

Recurrent Neural Network (RNN)

- There do exist connections backwards through the layers. Every connection introduces a form of memory into the network.
- Due to the fact that information does not flow in one direction through the network but **remains circulating inside** and is integrated with information from previous time steps. Because of that reason the memory exists within the network forms the network as a **dynamical system.**
- **The activation values of the neurons are not only determined by the current input but also by the previous state of network** (and thus, recursively, by all previous inputs.)

Recurrent Neural Network (RNN)

- An input history is kept within the internal state pathways, which are held in a nonlinear transformation matrices. This occurs due to the RNN possessing a dynamical memory. This allows the RNN to process temporal information through context.



(a) Feed forward Neural Network (b) Recurrent Neural Network

Figure 6: In recurrent neural network signals travel through loops. In feed-forward, signals only travel from input to output.

Recurrent Neural Network (RNN)

- The model proceeds information from the past, when recurrent connections are added, represented on Figure 7. Previous inputs can remain present in the dynamics of recurrent neural network and will affect the current output. The dynamical system learns to understand the relationships between **any past input** and the desired output.

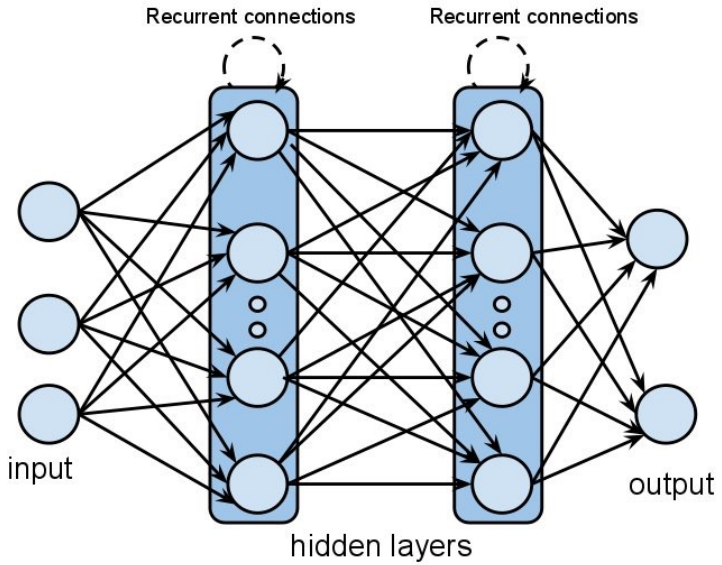


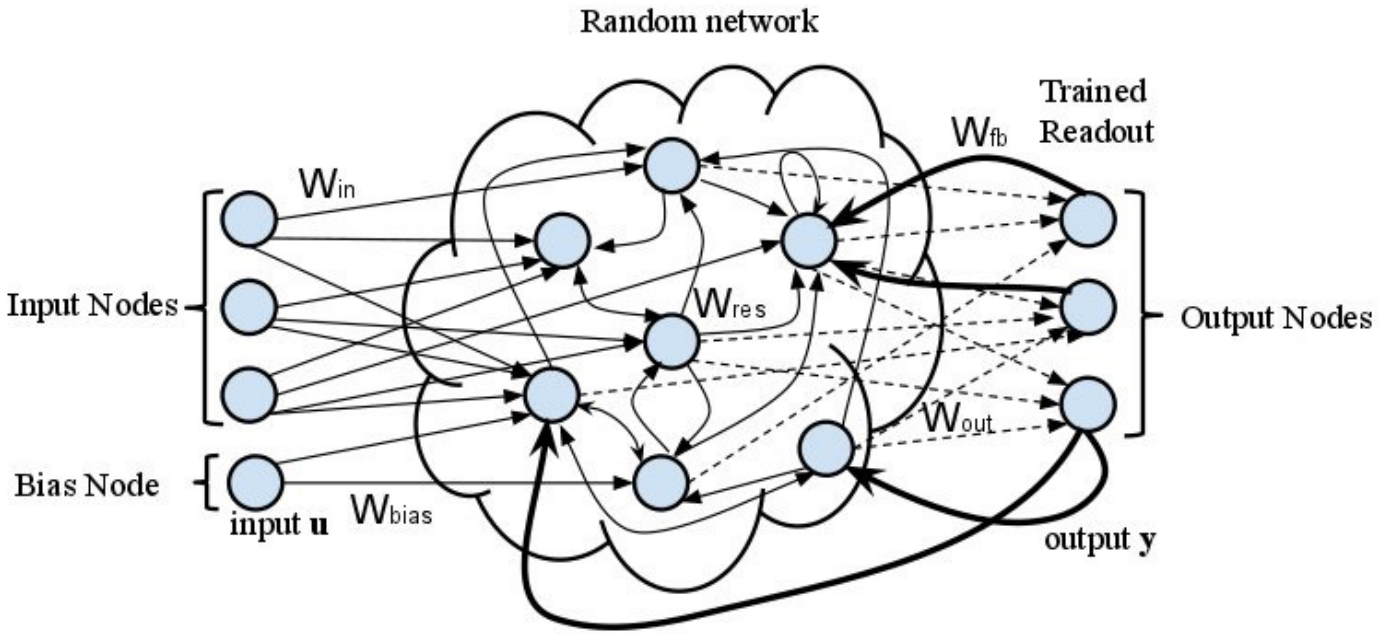
Figure 7: A schematic illustration of a feed-forward and recurrent neural network. The dashed lines are recurrent connections which are used here but normally not used in feed-forward neural networks.

Reservoir Computing (Echo State Networks)

- The Echo State Network (ESN) concept describes an engineering approach to **training** and **using** recurrent neural networks. ESN approach the RNN(called *reservoir*) is generated **randomly**, and only the readout from the reservoir is trained.
- ESN is an efficient training and using method for large RNNs.
- Due to the recurrent connections in the network, information about past inputs is stored in the network.
- *Echos:* The network contains a rich set of nonlinear transformation and mixings of the input signals of the current and past time-steps.
- Global parameter optimization is required for optimal dynamical regime.

Reservoir Computing (Echo State Networks)

- Role of the reservoir: complex nonlinear multidimensional filter that projects the input signals into a high-dimensional space, where the classification can be done much more accurately.



Global Parameters

- Main global parameters have massive impact on the performance of an ESN. Global Parameter: Echo State Network model is generated based on 4 parameters: **Spectral Radius, Bias Scaling, Output Feedback Scaling, Leak Rate.**

Parameter	Description	Range	Sample
N	Network Dimension	> 100	None
p	Spectral Radius	[0.5; 1.5]	Linear
β	Bias Scaling	[0.0; 1.0]	Linear
o	Output Feedback Scaling	[0.01; 10]	Logarithmic
λ	Leak Rate	[0.01; 1.0]	Logarithmic

Table 1: Global parameters and their common range.

Optimizing parameters

- To find the optimal unique combination of global parameters cross-validation is mostly applied. Because the reservoir weights are randomly constructed, this process is repeated for multiple reservoir initializations, often around ten independent initializations are used. The four global parameters that affect the reservoir system are **optimized by grid search**, which is computationally expensive and time consuming. Often the parameters are optimized one by one. In this study each parameter was optimized simultaneously using grid search.
  - Unique parameter combination is formed.

Optimized Global Parameters

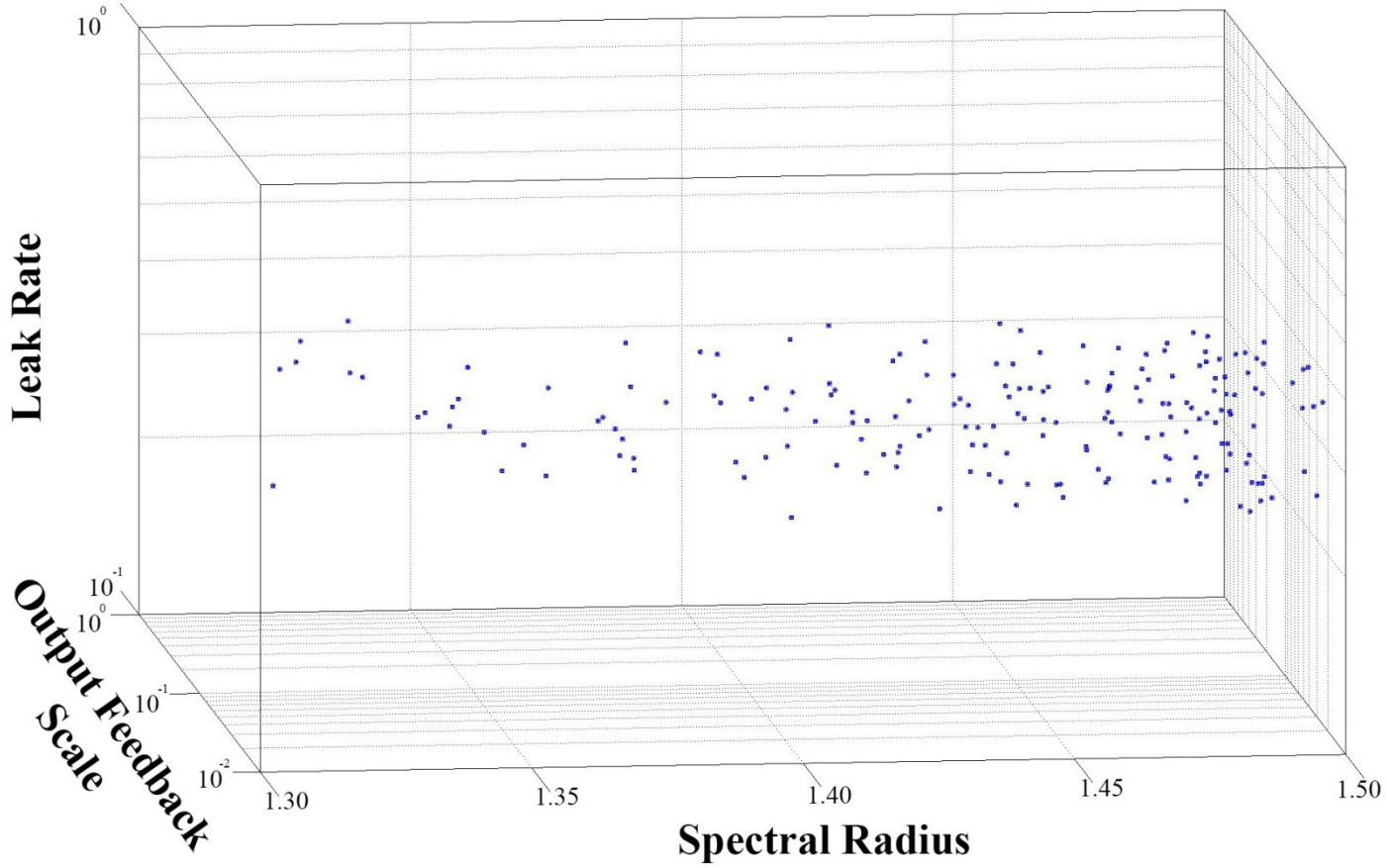


Figure *: Finalize grid search result, only accurate models' global parameter combinations are shown in blue points and all the other points were ignored since they were inaccurate or indicated by a straight line.

Typical Error Metrics

- In the research of among time series prediction, multiple error metrics have been proposed. In order to calculate the system's performance the parameters in *desired output* and *predicted output* was chosen such that an appropriate error function is minimized. NRMSE is use.
- Without normalization ESN will lose information and all neurons will be saturated. As a result, normalization should be done for every feature, which are input signals and independent from each other.

X = input signal traces for each feature

$$X' = X - \overline{x}$$

$$Xnorm = X' / \max(|X'|)$$

■ Normalized Root-Mean-Square Error (NRMSE), defined as:

$$NRMSE = \frac{1}{T} \sum_{target=1}^{T} \sqrt{\frac{(\hat{y}_{target} - y_{target})^2}{\sigma^2{}_y}}.$$

$\sigma^2{}_y$ expresses the variance of the original output y.

Typical Error Metrics

- Mean Square Error (MSE), defined as :

$$MSE = \frac{1}{T} \sum_{target=1}^{T} (\hat{y}_{target} - y_{target})^2.$$

with $\hat{y}_{target} - y_{target}$ the predicted and targeted value at time step *target*.

- Normalized Root-Mean-Square Error (NRMSE), defined as:

$$NRMSE = \frac{1}{T} \sum_{target=1}^{T} \sqrt{\frac{(\hat{y}_{target} - y_{target})^2}{{\sigma^2}_y}}.$$

${\sigma^2}_y$ expresses the variance of the original output y.

- Normalized MSE (NMSE), defined as: is dependent on the range of the data due to the division by the variance.

$$NMSE = \frac{1}{T} \sum_{target=1}^{T} \frac{(\hat{y}_{target} - y_{target})^2}{{\sigma^2}_y}.$$

- Mean Absolute Percentage Error (MAPE), defined as

$$MAPE = \frac{100}{T} \sum_{target=1}^{T} \left| \frac{\hat{y}_{target} - y_{target}}{y_{target}} \right|.$$

- *Symmetric* MAPE (SMAPE), defined as: is used very often (Makridakis and Hibon, 2000)

$$SMAPE = \frac{100}{T} \sum_{target=1}^{T} \frac{\hat{y}_{target} - y_{target}}{\frac{\hat{y}_{target} + y_{target}}{2}}.$$

Adapting ESNs for Short-term Load Time Series Prediction

- Evaluation of the overall detection performance is measured by, a unique combination of global parameters was trained and tested using complete training, complete test and a variety of validation sets consecutively. A schematic representation is shown in Figure3.1.
- Error metric selection.

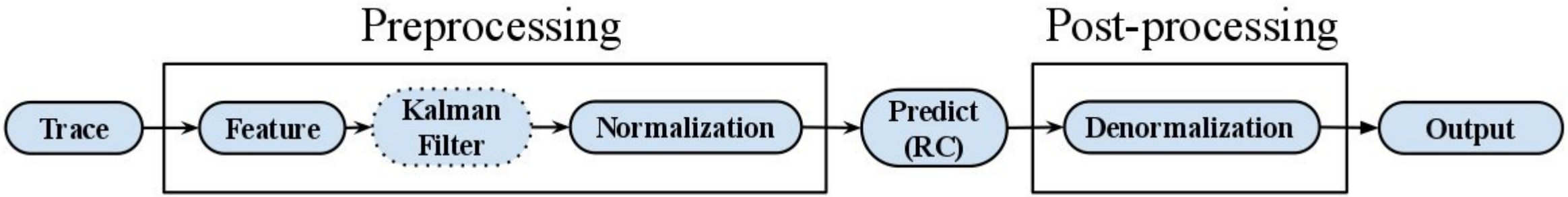


Figure 8: A schematic representation of baseline of the time series prediction scheme in this work.

Cross-Validation

- Cross-validation is used to select the optimal global parameter combinations.
- Of the Several cross-validation techniques that exist (Kohavi, 1995), the most frequently used is the n-fold cross-validation, where the original trace is subdivided in three parts: **training, validation and testing.**
- **Experiment percent for the Training , Validating and Testing**
  - Training: %80 of the trace.
  - Validating: %10 of the trace.
  - Testing(Unseen) Data: %10 of the trace.
- The larger the training sets usually allow us to take into account the information of past experiments which will increase the accuracy of our results. Despite this advantage, larger sets take longer calculation time. Training size could also be decreased, since many individual experiments will be ran.

Cross-Validation

- The following simple example explains this principle: a dataset consisting of 6 samples contains within each individual sample an equal number of data points, which are ordered sequentially.

	Sample1	Sample2	Sample3	Sample4	Sample5	Sample6
Fold1:	[%10 Validate]	[%20 Train]	[%20 Train]	[%20 Train]	[%20 Train]	[%10 Test set]
Fold2:	[%20 Train]	[%10 Validate]	[%20 Train]	[%20 Train]	[%20 Train]	[%10 Test set]
Fold3:	[%20 Train]	[%20 Train]	[%10 Validate]	[%20 Train]	[%20 Train]	[%10 Test set]
Fold4:	[%20 Train]	[%20 Train]	[%20 Train]	[%10 Validate]	[%20 Train]	[%10 Test set]
Fold5:	[%20 Train]	[%20 Train]	[%20 Train]	[%20 Train]	[%10 Validate]	[%10 Test set]
Final:	[%20 Train]	[%20 Train]	[%20 Train]	[%20 Train]	[%10 Train]	[%10 Test set]

Figure 9: The schematic representation of 5-fold cross-validation.

Tips I have used in my training process

- Look for a good error metric.
- If the goal is to predict n=180 steps ahead and the results came back inaccurate, then parallel smaller steps such as 10, 20, 30, 60 can give us better accuracy, from there the n value can be increased in small margins.
- The neuron size can be increased to gain additional performance and it is imperative to optimize the regularization parameter accordingly.
- Adding more features as inputs could increase the predictive power.

My Experiment Results

- On my work, my objective is to predict the workload for a short-term (three minutes ahead) prediction interval using the reservoir computing approach. This research is concentrated on understanding the differences between direct n-point ahead and recursive predictions using reservoir systems in short term predictions. The used trace was very aggressive in peaks and difficult to make short term predictions.

Experimental Results

Validation average performance	Peak detection percent
Detected peaks in range of ± 30	80.489
Detected peaks in range of ± 15	52.143
Detected miss peaks in range of ± 30	20
Detected miss peaks in range of ± 15	59

Table *: ESN model's validation performance

Test performance	Peak detection percent
Detected peaks in range of ± 30	75.532
Detected peaks in range of ± 15	47.872
Detected miss peaks in range of ± 30	11
Detected miss peaks in range of ± 15	45

Table *: ESN model's testing performance

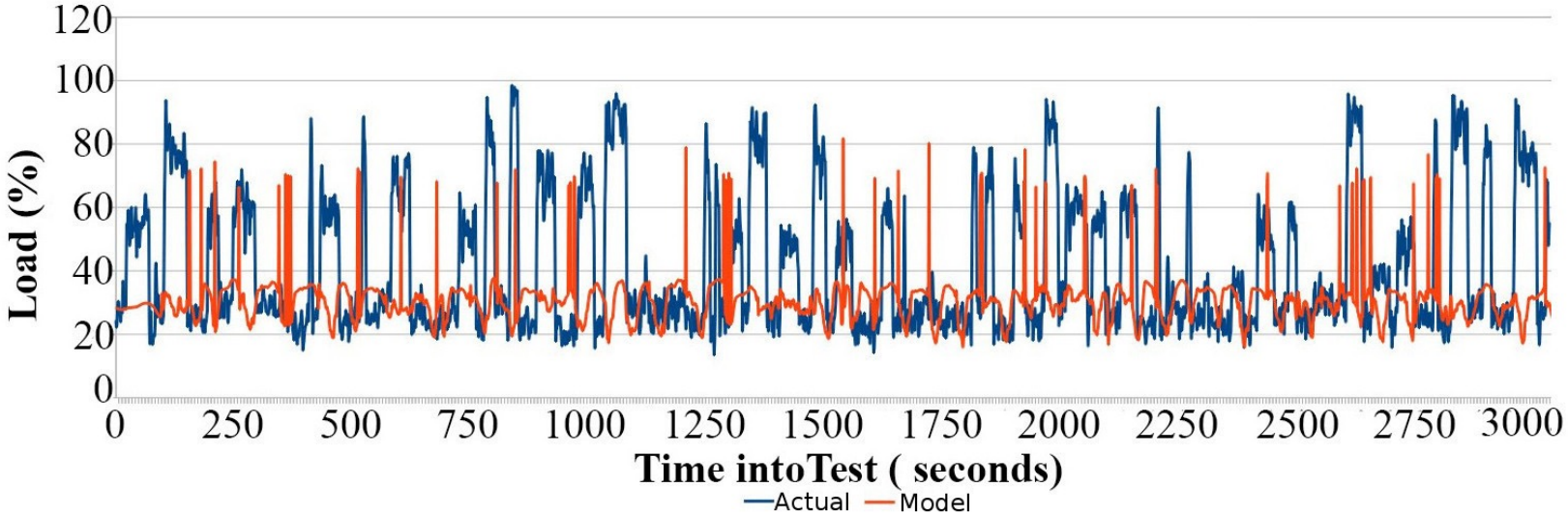


Figure 10: A simple example output of predictions on the untrained data. Red line is the predicted output and blue line is the desired output.

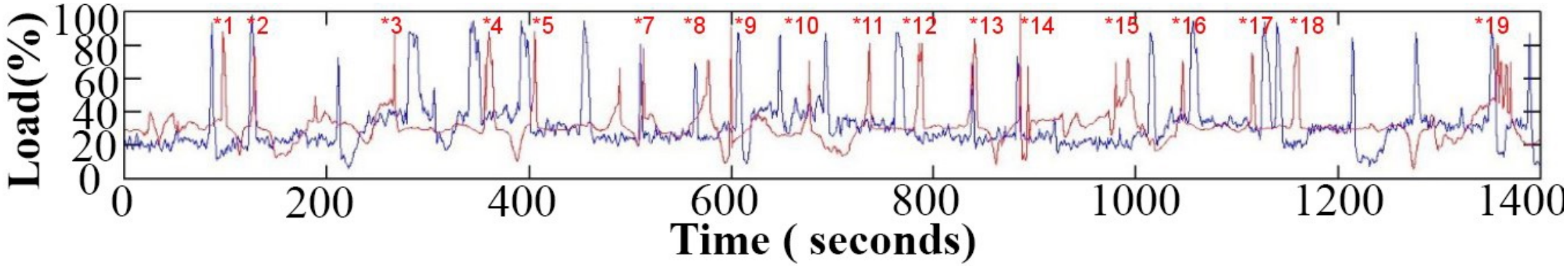


Figure 11: Example of peak detection for three minute ahead prediction. Red line is predicted valued and blue line is the original value.

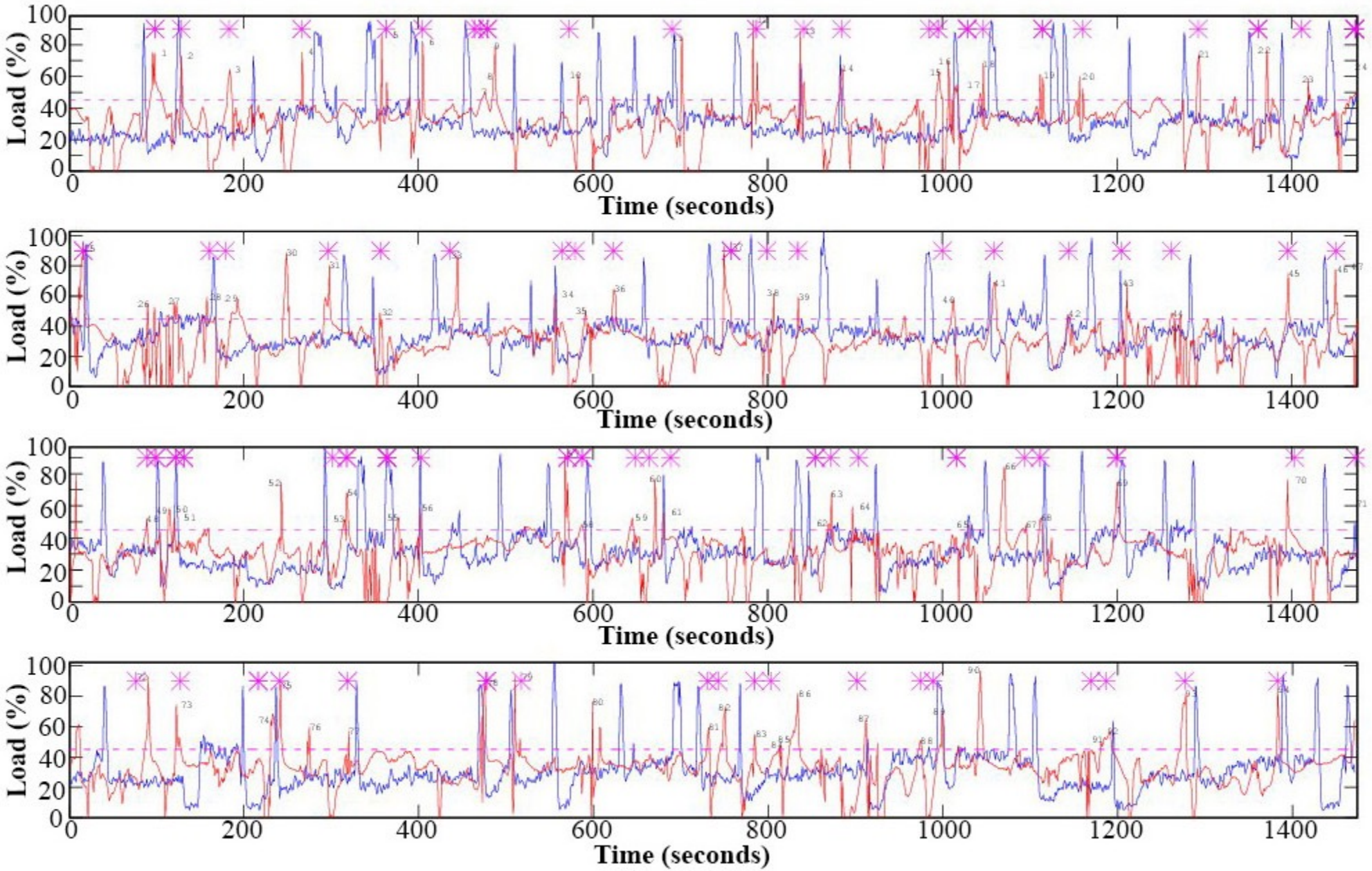


Figure 12: A schematic representation of a example output of predictions on the test data. This predictions are done by the model that is generated by the most success combination of the global parameters from 5-fold cross validation. Red line is the predicted output and blue line is the desired output. Purple stars identifies that the spikes is detected in the predicted spikes that are greater than 50 and the range of 30.

References

Simon, P. (2013). Too Big to Ignore: The Business Case for Big Data. Wiley Publishing, p.48, 1st edition.

Mitchell, T. (1997). Machine Learning. McGraw Hill, p.2.

Harnad, S. (2008). The annotation game: On turing (1950) on computing, machinery, and intelligence (published version bowdlerized). In Epstein, R., Roberts, G., and Beber, G., editors, Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer, pages 23–66. Springer. Chapter: 3 Commentary On: Turing, A.M. (1950) Computing Machinery and Intelligence. Mind 49 433-460 Address: Amsterdam.

Zadeh, L. A. (1994). Fuzzy logic, neural networks, and soft computing. Commun. ACM, 37(3):77–84.

Verstraeten, D. (2009). Reservoir Computing: computation with dynamical systems. PhD thesis, Ghent University.

Verstraeten, D., Dambre, J., Dutoit, X., and Schrauwen, B. (2010). Memory versus non- linearity in reservoirs. In IJCNN, pages 1–8. IEEE.

Verstraeten, D., Schrauwen, B., D'Haene, M., and Stroobandt, D. (2007). An experimental unification of reservoir computing methods.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report, 148:34.

Jaeger, H. (2002a). Adaptive nonlinear system identification with echo state networks. In Advances in neural information processing systems, pages 593–600.

Jaeger, H. (2002b). Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report, page 48.

References

Jaeger, H. (2002c). "tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the “echo state network” approach". Gmd-report, GMD - German National Research Institute for Computer Science.

Jaeger, H. (2008). A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the ”echo state network” approach. Technical report, Fraunhofer Institute for Autonomous Intelligent Systems.

Jaeger, H. and Haas, H. (2004). Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication.

Li, D., Han, M., and Wang, J. (2012). Chaotic time series prediction based on a novel robust echo state network. Neural Networks and Learning Systems, IEEE Transactions on, 23(5):787–799.

Legenstein, R. and Maass, W. (2007). 2007 special issue: Edge of chaos and prediction of computational performance for neural circuit models. Neural Netw., 20(3):323–334.

Zhang, G. P. and Qi, M. (2005). Neural network forecasting for seasonal and trend time series. European journal of operational research, 160(2):501–514.

Wyffels, F. (2013). Sequence generation with reservoir computing systems. PhD thesis, Ghent University.

Wyffels, F. and Schrauwen, B. (2010). A comparative study of reservoir computing strategies for monthly time series prediction. Neurocomput., 73(10-12):1958–1964.

Wyffels, F., Schrauwen, B., and Stroobandt, D. (2008). Stable output feedback in reservoir computing using ridge regression. In Proceedings of the 18th International Conference on Artificial Neural Networks, Part I, ICANN ’08, pages 808–817, Berlin, Heidelberg. Springer-Verlag.

Wyffels, F., Schrauwen, B., and Stroobandt, D. (2008). Using reservoir computing in a decomposition approach for time series prediction. In Lendasse, A., editor, Proceed- ings of ESTSP 2008 European Symposium on Time Series Prediction, pages 149–158. Multiprint Oy/Otamedia.