วารสารวิศวกรรมศาสตร์และนวัตกรรม
Journal of Engineering and Innovation

บทความวิจัย

Synthesis of training samples for the online sequential extreme learning machine and application in load forecasting

Charnon Chupong Boonyang Plangklang*

Department of Electrical Engineering, Faculty of Engineering, Rajamangala University of Technology Thanyaburi, Pathum Thani 12110

* Corresponding author.

E-mail: boonyang.p@en.rmutt.ac.th; Telephone: 0 2549 3425

Received 18 August 2022; Revised #1 24 December 2022; Accepted 1 March 2023

Abstract

Online Sequential Extreme Learning Machine (OS-ELM) is a model capable of incremental learning from newly received samples while working, but getting start with the OS-ELM requires sufficient and appropriate sample data for initial training. In some cases, finding such sample data is not possible. To address this issue, this article presents a synthesis of sample data for the initial training of the OS-ELM model. The proposed method is to take the first sample at the time of OS-ELM initialization and then add noise to transform it to be new sufficient samples. In this article, the authors have compared different formats of the noise used in the synthesis of the sample data. It was found that the Gaussian noise with properly selected the standard deviation value gives training samples that helped the OS-ELM forecast load with the most accuracy. In addition, the use of uniform noise allows the OS-ELM to have slightly lower accuracy than Gaussian noise but can be used without worrying about selecting the appropriate standard deviation value.

Keywords

incremental learning; load forecasting; OS-ELM; training sample

1. Introduction

Sample data is very important in machine learning due to the appropriate and sufficient sample data is required to train the model. In some cases, the model has been used for a while and the environment has changed, this makes the old sample data used to train the model unsuitable for the new environment. For example, the model for load forecasting uses past sample data to train, but over time, the behavior of loads has changed dramatically, causing the sample data that was used to train the model inconsistent with the change resulting in significant error in forecasting. To solve this problem, a new set of sample data was needed to train the model again, but such an approach takes time and costs, causing practical inconvenience. The researchers developed some methods that allow the model to learn more from the newly received sample data without forgetting the already learned sample data known as "incremental learning" or "online learning".

There are some incremental models/algorithms that are frequently mentioned in research papers, such as Incremental Learning Vector Quantization (ILVQ) [1], Incremental Support Vector Machine (ISVM)

[2], Learn++ [3], Stochastic Gradient Descent (SGD) [4], Online Random Forecast (ORF) [5], and Online Sequential Extreme Learning Machine (OS-ELM) [6]. In this article, the authors focus on the OS-ELM as it is a model that can be used in both classification and regression tasks and can also learn quickly from the input with a low computational cost which can be run on low computing power devices.

The structure of OS-ELM is like an Artificial Neural Networks (ANN) with a single hidden layer, but its learning is unlike the ANN. The learning of OS-ELM does not use an iterative approach such as gradient descent, but uses random weights and calculated weights by a recursive least square method. In other words, the weights between the input layer and the hidden layer are randomly assigned and have a constant value over time and the weights between the hidden layer and the output layer use the Recursive Least Square method to calculate the value every time a new sample is received.

Although OS-ELM can incrementally learn from the newly received sample data, but getting start with OS-ELM requires a certain amount of sample data to initially train the model. The number of this initial sample data must not be less than the number of nodes in the hidden layer [6] and the number of nodes in the hidden layer also affects the model performance [7]. Therefore, it can be said that the amount of initial training sample data affects the performance of the model. This is a limitation to the use of OS-ELM if sufficient and accurate sample data cannot be obtained, e.g., load forecasting in new buildings or areas where electricity usage has never been recorded.

Several studies have proposed a solution for the problem of insufficient training samples. For example, using data pooling for individual household load forecasting [8-9], which collects datasets from other relevant sources and finds the suitable method to choose some samples for training the model. To use such an approach, one still has to collect some datasets to use for training anyway. In [10], the application of the Generative Adversarial Network (GAN) was presented to synthesize data on electric vehicle charging. GAN is a deep learning model, which is not suitable for use in systems with low computational resources. In [11], some data synthesis methods for training wind turbine power forecasting models were presented. One proposed method combines existing datasets with Gaussian distribution noise to increase the number of samples used to train the LSTM model. In [12], load forecasting was presented using the OS-ELM model. The initial training samples for the model were obtained by using the data from the first measure and added with a uniform distribution noise to increase the number of samples. In this approach, a load forecasting model can be used in a system with low computational resources and can be used online without needing any dataset.

The method presented in [12] uses uniform-distribution noise without considering other distribution forms of noise. Therefore, in this article, we present a method of sample data generation for initially training the OS-ELM by adding noise with different PDFs as shown in Fig. 1. The following contents are divided as follows: related works in part 2, methodology in part 3, experiment and results in part 4 and part 5 is discussion and conclusion.

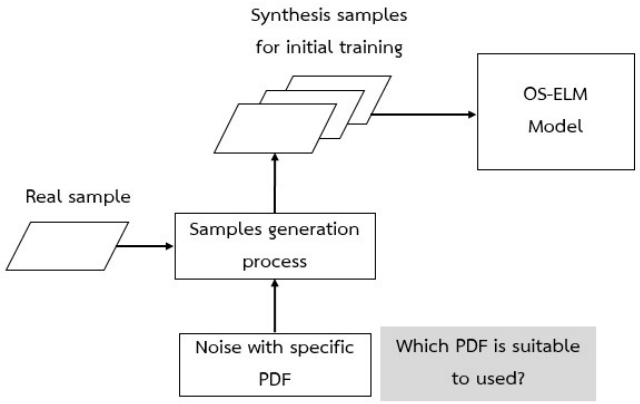

Fig.1 Main point of this article

2. Related Works

2.1 Some incremental models/algorithms

There are some incremental learning models/ algorithms often mentioned in various research studies as follows:

- Incremental Learning Vector Quantization (ILVQ) [1] is an incremental version of Learning Vector Quantization (LVQ) that uses the principle of Prototyped-based learning [13]. In the algorithm, it checks whether the received data is different from the past data that has been learned or not if significant differences are found, a new prototype will be created. This allows the model to work with changed data without forgetting the past data.
- Incremental Support Vector Machine (ISVM) [2], is like a Support Vector Machine (SVM), but some incoming data are recorded and called the Candidate Vector. The Candidate Vector may be set as Support Vector depending on the difference between the newly received data and the existing Support Vector.
- Learn++ [3] uses the ensemble model principle as Ada-boost [14], consisting of sub-models created by learning from past data. Past data where the model failed to perform is more likely to be chosen to train the sub-model. Therefore, it is said that the model can learn from mistakes to work with the changing data.
- Stochastic Gradient Descent (SGD) [4] It is an iterative method for adjusting the model parameters to optimize an objective function. Each parameter adjustment can use only a fraction of the sample data to be trained, so SGD can be applied as incremental learning.
- Online Random Forest (ORF) [5] is a Random Forecast model when it is found that the input data is different from the past data that has already been learned. The model will increase the number of trees to work with this data. Thus, ORF can work with changing data without forgetting the past data.
- Online Sequential Extreme Learning Machine (OS-ELM) [6] is an incremental version of Extreme Learning Machine (ELM) [15] which is characterized by extremely fast learning speed and low computation cost. The details of OS-LEM will be discussed in the next section.

2.2 Online Sequential Extreme Learning Machine (OS-ELM)

OS-ELM was introduced by N.Y. Liang in 2006, it is based on ELM and adds incremental learning capabilities. The structure of OS-ELM is the same as Feed-Forward Artificial Neural Networks as shown in Figure 2.

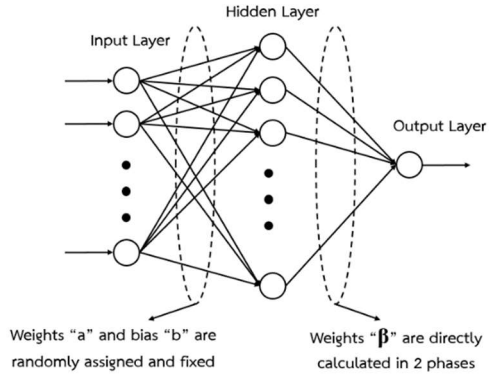

Fig.2 Structure of the OS-ELM

The weights between the input layer and the hidden layer are assigned randomly and remain constant over time. The weights between the hidden layer and the output layer are calculated in two phases as follows:

1) Initial training phase: For some N samples $(\mathbf{x_j}, \mathbf{y_j})$ and hidden layer has L nodes. The relationship between $\mathbf{x_j}$ and $\mathbf{y_j}$ can be described as follows:

$$\mathbf{y}_j = \sum_{i=1}^{L} \boldsymbol{\beta}_i g(\mathbf{a}_i \mathbf{x}_j + b_i), \quad j = 1,2,3,...N \qquad (1)$$

where $\mathbf{a}_i \in \mathbb{R}^n$ are the weights in the input layer and $b_i \in \mathbb{R}$ are the bias in the input layer, $g(...): \mathbb{R} \to \mathbb{R}$ is an activation function in the hidden layer, and $\boldsymbol{\beta}_i \in \mathbb{R}^m$ are the weights in the hidden layer. Equation (1) can be simplified by writing it as follows:

$$\mathbf{Y} = \mathbf{H}\boldsymbol{\beta} \qquad (2)$$

where

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{a}_1 \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{a}_L \mathbf{x}_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(\mathbf{a}_1 \mathbf{x}_N + b_1) & \cdots & g(\mathbf{a}_L \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L \times M} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^T \\ \vdots \\ \mathbf{y}_N^T \end{bmatrix}_{N \times m}$$

The initial training is to calculate the β value as in equation (3), where $\mathbf{x}$ and $\mathbf{Y}$ are known from training samples data.

$$\boldsymbol{\beta} = \mathbf{K}^{-1}\mathbf{H}^T\mathbf{Y} \qquad \text{where } \mathbf{K} = \mathbf{H}^T\mathbf{H} \qquad (3)$$

If $\mathbf{x}$ and $\mathbf{Y}$ in the initial training samples have small amount or have low dimensions, the β value calculated using the normal equation as in equation (4) will be faster than using the gradient descent method [16]. Numerous studies have demonstrated that this learning method performs as well as other learning algorithms, although the input layer uses random weights and bias values [17-18].

This method may encounter the numerical instability problem when $\mathbf{H}^T\mathbf{H}$ is non-invertible. To solve this problem, two main solutions were proposed: the first solution is using the Singular Value Decomposition (SVD) method to inverse the matrix $\mathbf{H}^T\mathbf{H}$ [19]. The second solution is to use a regularization factor adding to the matrix $\mathbf{H}^T\mathbf{H}$ before inversion as shown in equation (4) [20].

$$\mathbf{K} = \mathbf{H}^T\mathbf{H} + \lambda\mathbf{I} \qquad (4)$$

where λ is a very small value called the regularization factor and $\mathbf{I}$ is the identity matrix**.**

2) Incremental learning phase: This is the process for adjusting the β value to be suitable to the newly received sample data as well as the previously learned data. The incremental learning phase uses the principle of the recursive least square method. The updated beta value is a function of the previous beta value recursively as in equations (5) and (6) [6].

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mathbf{K}_{k+1}^{-1}\mathbf{H}_{k+1}^T(\mathbf{Y}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}_k) \qquad (5)$$

where

$$\mathbf{K}_{k+1} = \mathbf{K}_k + \mathbf{H}_{k+1}^T\mathbf{H}_{k+1} + \lambda\mathbf{I} \qquad (6)$$

The subscription terms k and $k+1$ refer to the sample data in k^{th} order and the sample data in $(k+1)^{th}$ order, respectively. Where $k = 0$ means the value from the initial training phase.

2.3 OS-ELM without initial training sample

The use of OS- ELM requires sufficient and appropriate sample data to be used in the initial training phase. In some cases where sample data cannot be obtained, for example, load forecasting in new buildings or areas that have never collected electricity usage data, this makes the use of OSELM limited.

To solve this problem reference [12] proposed a method to use the OS- ELM without using the initial training sample data for load forecasting application. This method uses the first load profile sample obtained at startup. This sample is added with some noise value to create enough synthesis samples for initial training as shown in Fig.3.

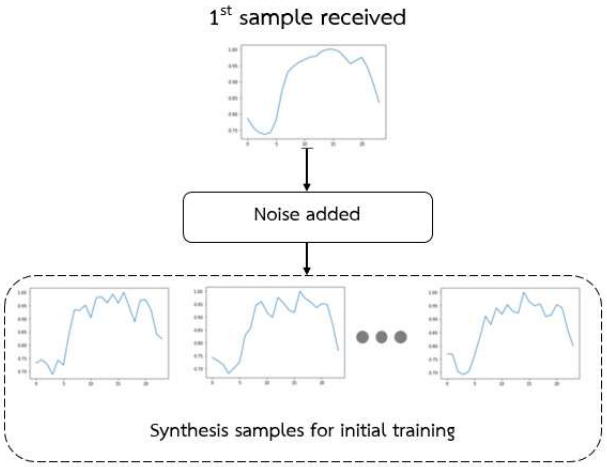

Fig.3 Synthesizing new load profile samples by adding noise value

3. Proposed Method

The synthesis load profile sample for initial training of the OS- ELM presented in section 2. 3 is created by noise with a uniform probability density function (pdf). The article in section 2. 3 did not experiment with other pdf of noise, such as gaussian, which is commonly used to create augmented data for training the deep learning model. Therefore, this article proposed the use of gaussian noise compared to the use of Uniform noise to synthesize sample data for initial training of the OS- ELM. The algorithm for synthesizing the sample data is shown in Fig.4.

Algorithm: Synthesize data by noise adding

INPUT: *data* = 1st sample in dataset that is scaled to value between(0,1)
N = numbers of output sample required
PDF = Noise probability density function

OUTPUT: *synt_data*

STEP:
1: **for** *n=0* **to** *n=N:*
2: **for** *i*=0 **to** *i*=length(*data*):
3: *rand[i]* = random value according to PDF
4: *synt_data[n][i] = data[i] + (data[i]*0.1)*rand[i]*
5: **end for**
6: synt_data[n]/max(synt_data[n])
7: **end for**
8: **return** *synt_data*
9: **end**

Fig.4 Synthesizing data by noise adding algorithm

From the algorithm in Fig. 4, it can be seen that the data used to synthesize the training samples is scaled to a value between 0 and 1 and the noise pattern is set by the user. This article uses a uniform pdf of noise with a value between 0 and 1 as in reference [12]. The synthesis process uses uniform noise compared with gaussian noise with a mean of 0 and a standard deviation (std.) of 0.1, 0.5, and 1. The noise pdfs used in this article are shown in Fig. 5 and the examples of load profiles data synthesized by that noise are shown in Fig.6.

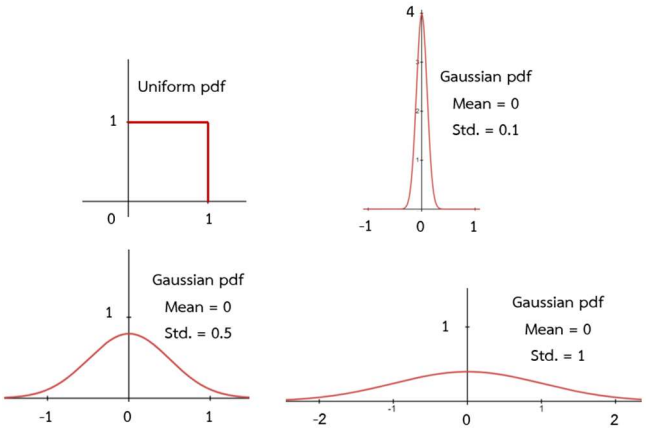

Fig.5 Probability density function (pdf) of noise used to synthesize the training samples

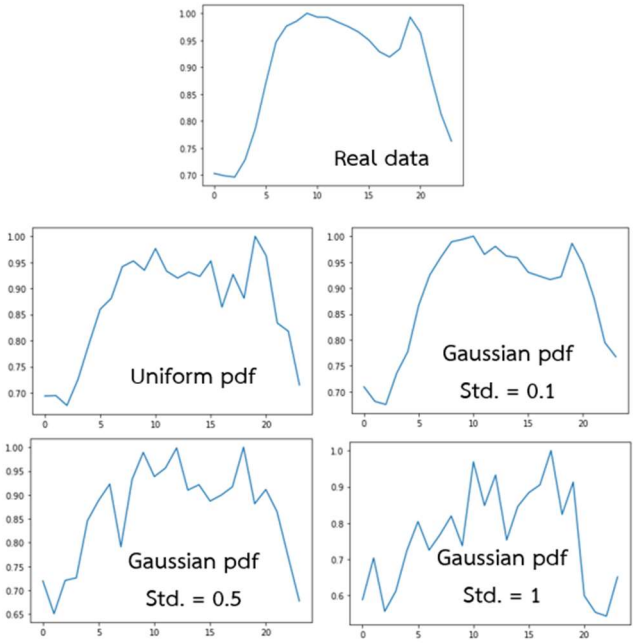

Fig.6 Example of load profile synthesized by each pdf of noise

4. Experimental

The experiment in this article used a dataset called "Hourly Energy Consumption" from Kaggle.com [21], which is an hourly load profile collected from 9 utility companies in the eastern United States. The dataset will be used as sample data for OS-ELM to learn. Each sample is divided into two parts: the input part and the target part. The target part is the hourly load and the input part is the 24-hourly load value before the target as shown in Fig. 7. We chose a 24 hours time lag as input because it has a high autocorrelation value [22].

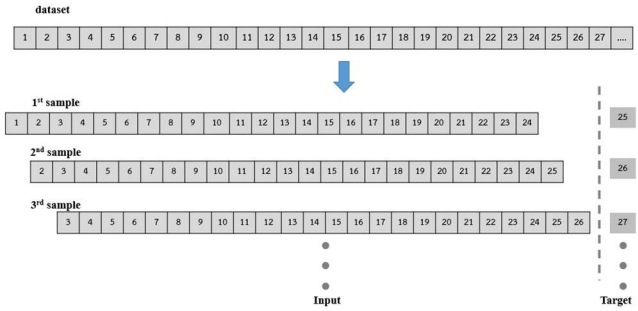

Fig.7 Input and Target from the dataset

The experimental process starts with synthesizing the sample data for initial training OS-ELM based on the algorithm in Fig.4. Then the OS-ELM forecasts the next hour's load using the previous 24 hours load as input. The forecast load values and the actual load values in the dataset are compared. The mean absolute percentage error (MAPE) is used as an indicator of OS-ELM performance. The experiment is repeated in the following sample data as shown in Fig. 8. The initial training samples for OS-ELM in this experiment use noise with uniform pdf and gaussian pdf with std. equal to 0.1, 0.5, and 1. The experiment is performed on 72 samples (72 hours) to evaluate the performance of OS-ELM when startup.

This experiment uses the OS-ELM model with 24 input nodes equal to the dimension of the input data, 1 output node equal to the dimension of the output data, and a hidden layer of 50 nodes obtained from the experimental results shown in Fig 8. Fig. 8 was the result of using the OS-ELM model with different numbers of hidden layers were used to forecast the load of AEP data, and we found 50 nodes in the

hidden layer provide the appropriate accuracy and timing.

To improve the accuracy of forecasting, this experiment also uses an ensemble model of 10 OS-ELM models derived from the experimental results shown in Fig. 9. Fig. 9 was a result of using different numbers of models to forecast the load of the AEP dataset. We found that 10 models were the appropriate accurate and time- optimized results. That is, the final forecast value was derived from the mean of all 10 models of forecast values as shown in Fig. 11.

The synthesized algorithm in Fig. 4 should generate 50 sets of sample load profiles as the number of nodes in the hidden layer, that is the minimum samples for initial learning [6].

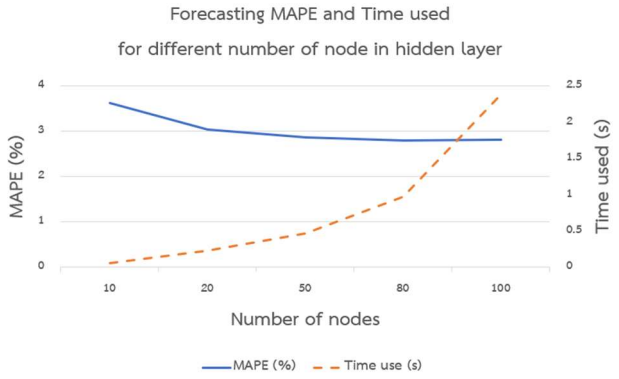

Fig. 8 MAPE and time used for different numbers of the node in a hidden layer

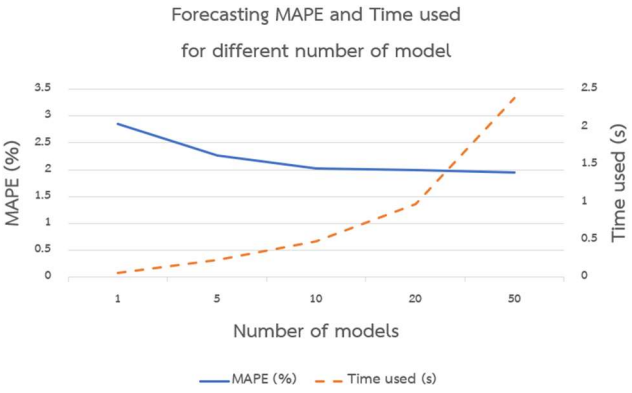

Fig. 9 Forecasting MAPE and time used for different numbers of model

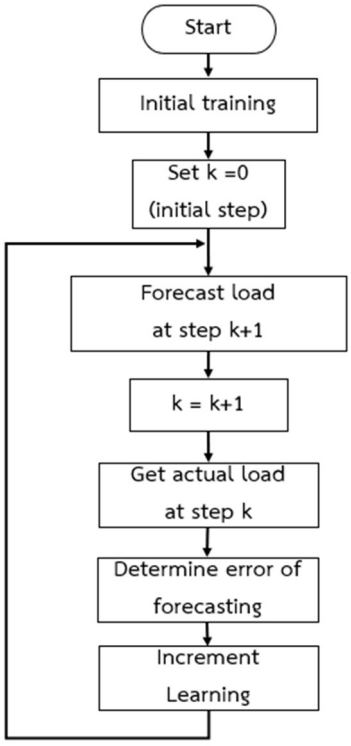

Fig.10 Experiment process

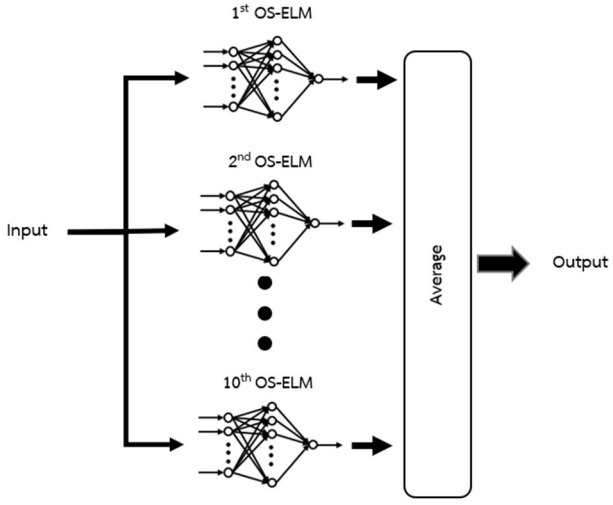

Fig.11 Ensemble of 10 OS-ELM models

5. Results and Discussion

Table 1 shows the MAPE results of load forecasting of the 9 utility datasets using different noise pdfs, and the bottom row of the table shows the average values of MAPE from all datasets.

Table 1 Experimental results

Dataset	MAPE when using each pdf of noise (%)			
	Uniform	Gaussian Std.=0.1	Gaussian Std.=0.5	Gaussian Std.=1
AEP	2.03	**1.95**	2.11	2.44
COMED	1.61	**1.47**	1.80	2.19
DAYTON	2.73	2.80	**2.65**	2.89
DEOK	1.99	**1.83**	2.21	2.52
DOM	1.99	2.07	**1.95**	2.36
DUQ	2.53	3.10	**2.44**	2.44
EKPC	3.67	4.37	3.13	**2.93**
FE	**2.31**	2.67	2.38	2.57
NI	1.63	1.70	**1.61**	1.92
Average	2.28	2.44	**2.25**	2.47

From Table 1, if considering each dataset, the most accurate is gaussian pdf noise with a standard deviation equal to 0. 5 gives the most accurate forecast for the four datasets: DAYTON, DOM, DUQ, and NI. The second most accurate is gaussian pdf noise with a standard deviation equal to 0. 1 which yielded the most accurate forecast for 3 datasets: AEP, COMED, and DEOK. Suppose the average MAPE of all 9 datasets is considered, the gaussian pdf noise with std. equal to 0.5 still the most accurate with an average MAPE of 2.25%, followed by the uniform pdf noise with an average MAPE of 2.28%.

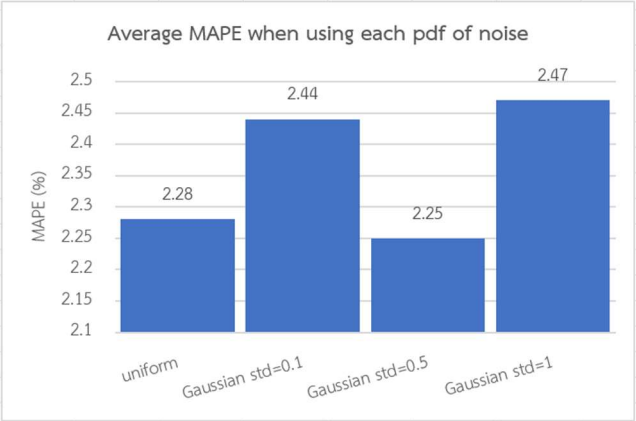

Fig.12 Average MAPE for each pdf of noise

Figure 13-16 shows some comparison graphs of forecasted load values and actual load values in the AEP data set when using different PDFs of noise.

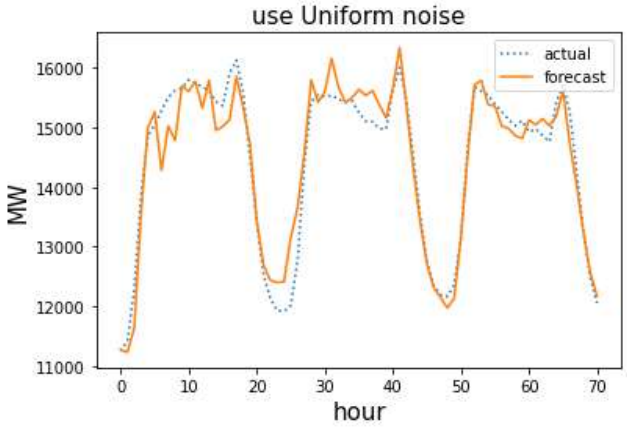

Fig.13 Forecasted load and actual load when using uniform noise

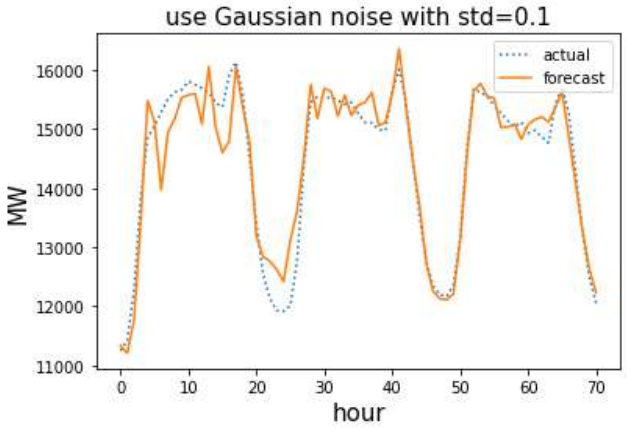

Fig.14 Forecasted load and actual load when using Gaussian noise with std = 0.1

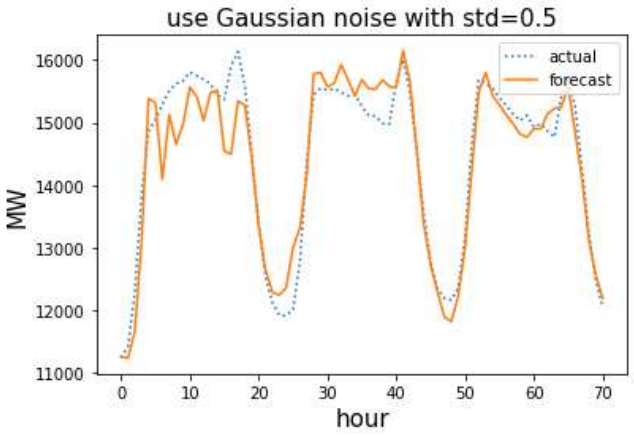

Fig.15 Forecasted load and actual load when using Gaussian noise with std = 0.5

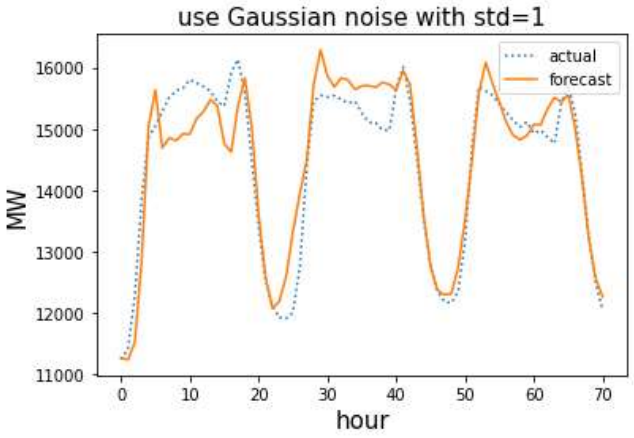

Fig.16 Forecasted load and actual load when using Gaussian noise with std = 1

This experiment found that using gaussian pdf noise to synthesize the initial training dataset for the OS-ELM would allow the most forecasting accuracy. To achieve the most forecasting accuracy, an appropriate standard deviation must be selected. In this case, the standard deviation of 0.5 is selected meaning there is a 95% probability that the noise will range between -1 and 1. In this experiment the load profile data is scaled to a range of 0 to 1, then Gaussian pdf noise with a standard deviation of 0.5 is appropriate to synthesize the training data.

The use of uniform pdf noise to synthesize the initial training dataset resulted in slightly low forecasting accuracy than using a Gaussian pdf noise with 0.5 standard deviations. However, the use of the uniform pdf noise still had higher forecasting accuracy than the gaussian pdf noise with standard deviations of 0.1 and 1. The reasons are as follows: Gaussian pdf noise with the standard deviation of 0.1 has low dispersion, making the synthesized sample data look similar, causing the model to have over-fit problems. Gaussian pdf noise with the standard deviation of 0.5 has high dispersed making synthesized sample data look different that it can't represent the real data, causing the model to have under-fit problems.

6. Conclusion

In conclusion, choosing a noise to create a dataset for the initial training of the OS-ELM requires an appropriate pdf pattern. In this experiment, the gaussian pdf noise with a standard deviation of 0.5 and the uniform pdf noise are the appropriate pdf pattern. If narrowly dispersed noise is selected, the synthesis samples will be very similar, causing the model to have an over-fit problem. If wide dispersion noise is selected, the synthesis sample will differ too much from the actual data, causing the model to have an under-fit problem.

Reference

[1] Sato A, Yamada K. Generalized learning vector quantization. In: *Proceedings of the 1995 Neural*

Information Processing Systems. Denver, Colorado, USA; 1995. p.423-429.

[2] Cauwenberghs G, Poggio T. Incremental and decremental support vector machine learning. *Advance Neural Information Processing Systems*. 2001(13): 388-394.

[3] Polikar R, Upda L, Upda S, Honavar V. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 2001;31(4): 497-508.

[4] Zhang T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. Banff, Alberta, Canada; 2004. p.116-123.

[5] Saffari A, Leistner C, Santner J, Godec M, Bischof H. On-line random forests. In*: Proceedings of the 2009 IEEE Twelfth International Conference on Computer Vision Workshops*. Kyoto, Japan; 2009. p.1393-1400.

[6] Liang N, Huang G, Saratchandran P, Sundarrajan N. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transaction on Neural Networks*. 2006;17(6) : 1411-1423.

[7] Liu Y, Cao W, Liu Y, Zou W. A Novel Ensemble Learning Method for Online Learning Scenarios. In: *2021 IEEE 4th International Conference on Electronics Technology (ICET)*. Chengdu, China; 2012. p. 1137-1140.

[8] Shi H, Xu M, Li R. Deep Learning for Household Load Forecasting - A Novel Pooling Deep RNN. *IEEE Transactions on Smart Grid*. 2018;9(5): 5271-5280.

[9] Yang E, Youn C. Temporal Data Pooling with Meta-Initialization for Individual Short-Term Load Forecasting. *IEEE Transactions on Smart Grid*. 2022;14(4): 3246-3258.

[10] Xiaodong S, Houxiang Z, Yue X, Peng L, Junyong L. Short- term electric vehicles charging load forecasting based on deep learning in low-quality data environments. *Electric Power Systems Research*. 2022;212: 1-14.

[11] Hao C, Yngve B, Qixia Z. Data- augmented sequential deep learning for wind power forecasting. *Energy Conversion and Management*. 2021;258: 1-12.

[12] Charnon C, Boonyang P. Incremental learning model for load forecasting without training sample. *CMC-Computers Materials & Continua.* 2022;72(3): 5415-5427.

[13] Schwabacher M, Hirsh H, Ellman T. Learning prototype- selection rules for case- based iterative design. In: *Proceedings of the Tenth Conference on Artificial Intelligence for Applications.* San Antonio, TX, USA;1994. p. 56-62.

[14] Freund Y, Schapire R. A decision- theoretic generalization of on- line learning and an application to boosting. *Journal of Computer and System Sciences*. 1997;55(1): 119-139

[15] Huang G, Zhu Q, Siew C. Extreme learning machine: a new learning scheme of feedforward neural networks. In: *2004 IEEE International Joint Conference on Neural Networks*. Budapest;2004. p.985-990, 2004.

[16] Andrew ng. Machine learning specialization. Available from: https/ / coursera. org/ specializations/ machine- learning- introduction [Accessed 24th December 2022].

[17] Liu X, Lin S, Fang J, Xu Z. Is extreme learning machine feasible? a theoretical assessment (part I). *IEEE Transactions on Neural Networks and Learning Systems*. 2015;26(1): 7-20.

[18] Lin S, Liu X, Fang J, Xu Z. Is extreme learning machine feasible? A theoretical assessment (Part II). *IEEE Trans. Neural Networks Learning System.* 2015;26(1): 21-34.

[19] Klema V, Laub A. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control.* 1980;25(2): 164-176.

[20] Deng W, Zheng Q, Chen L. Regularized extreme learning machine. In: *2009 IEEE Symposium on Computational Intelligence and Data Mining.* Nashville, TN, USA; 2009. p. 389-395.

[21] R Mulla. Hourly Energy Consumption. Available from: https://kaggle.com/ robikscube/hourly-energy-consumption [Accessed 24th December 2022].

[22] Surakhi O, Zaidan M, Fung L, Hossein N, Serhan S, AlKhanafseh M, Ghoniem M, Hussein T, Time-Lag Selection for Time-Series Forecasting Using Neural Network and Heuristic Algorithm. *Electronics*. 2021; 10(20): Available