2022년 2월

석사학위 논문

Development of Diagnostic and Prognostic Algorithms for Nuclear Power Plant Using Artificial Neural Network

조선대학교 대학원

원자력공학과

김효진

Development of Diagnostic and Prognostic Algorithms for Nuclear Power Plant Using Artificial Neural Network

-인공신경망을 이용한 원자력 발전소 진단 및 예측 알고리즘 개발-

2022년 2월 25일

조선대학교 대학원

원자력공학과

김효진

Development of Diagnostic and Prognostic Algorithms for Nuclear Power Plant Using Artificial Neural Network

지도교수 김 종 현

이 논문을 공학 석사학위신청 논문으로 제출함

2021년 10월

조선대학교 대학원

원자력공학과

김 효 진

김효진의 석사학위논문을 인준함

위원장 조선대학교 교수 나만균 (인)

위원 조선대학교 교수 김종현 (인)

위원 조선대학교 교수 송종순 (인)

2021년 12월

조선대학교 대학원

Table of Contents

List of Tables

List of Figures

ABSTRACT

Development of Diagnostic and Prognostic Algorithms for Nuclear Power Plant Using Artificial Neuranl Network

Hyojin Kim
Advisor : Prof. Jonghyun Kim, Ph.D.
Department of Nuclear Engineering
Graduate School of Chosun University

비정상 및 비상 상황 발생 시 운전원은 상황을 인식하고 적절한 조치를 수행해야 한다. 비정상 상황에서는 운전원이 올바른 비정상 절차서를 수행하기 위하여 빠르고 정확한 진단을 수행해야 한다. 하지만, 원자력 발전소에는 100개 이상의 비정상 절차서와 약 4,000개의 경보 시스템으로 인해 운전원이 의사결정 과정에서 고려해야 할 정보가 매우 많다. 이러한 과도한 정보는 운전원에게 혼란을 줄 수 있을 뿐만 아니라 인적 오류를 유발할 수도 있다. 또한, 비상 상황에서는 운전원의 올바른 상황인식은 원자력 발전소 관리에 중요할 뿐만 아니라 사고를 효과적으로 완화하는 데 도움이 된다. 특히, Ensley가 제시한 3 Level 상황인식 중 Level 3는 (미래 상황 예측) 안전을 확보하기 위해 가장 좋은 조치를 결정하는데 필요한 지식 및 시간을 제공할 수 있지만, 원자력 발전소의 복잡성과 사고의 불확실성으로 인해 어려운 과제이다. 본 연구에서는 비정상 상황에서의 진단과 비상 상황에서의 예측을 돕기 위하여 인공신경망을 이용한 진단 및 예측 알고리즘을 제안한다. 진단 알고리즘은 long short-term memory (LSTM)과 variational autoencoder (VAE)를 이용하여 빠르고 정확한 진단을 할 뿐만 아니라 알지 못하는 사고를 식별할 수 있으며,

진단 결과를 검증하여 진단 신뢰도를 향상했다. 예측 알고리즘은 bidirectional LSTM (Bi-LSTM)과 attention mechanism (AM)을 이용하여 120단계를 통해 2시간 거동을 예측하고 variational encoder-decoder를 이용하여 예측의 불확실성을 제공한다. 제안된 알고리즘들은 compact nuclear simulator (CNS)를 이용하여 구현 및 검증하였다.

Ⅰ. Introduction

Owing to the complexity of engineering systems and potential hazards associated with nuclear power plants (NPPs), appropriate decisions must be ensured for their safe and efficient operation [1]. Most decisions are made by human operators who monitor numerous instrumentation signals and apply them to various procedures that correspond to the plant status [2]. In particular, if abnormal or emergency situations occur during the operation of NPPs, operators should be aware of the situation and execute appropriate actions. However, operators are inclined to perform incorrect measures owing to the complexity of accidents and time constraints involved [3].

Diagnosing NPPs in abnormal situations is considered one of the most difficult tasks of operators. First, there is excess information to consider in the decision-making process of operators. Moreover, NPPs not only have approximately 4000 alarms and monitoring devices in the main control room, but they also have more than 100 operating procedures that should be followed during abnormal situations [4]. Using this large body of information, operators diagnose abnormal situations and execute specific actions in accordance with the relevant operating procedures [5]. However, this excessive information could confuse operators and increase the likelihood of error caused due to excess mental workload [6]. Additionally, some abnormal situations require speedy diagnosis and response to prevent the reactor from being tripped.

Moreover, in emergency situations, adequate situation awareness (SA) of operators is not only important for the management of NPPs, but can also assist the operators in effectively mitigating events. Among the three levels of SA suggested by Ensley, Level 3 SA (i.e., projection of future status of the situation) provides the knowledge (and time) necessary to decide on the most favorable course of action, to ensure safety [7]. In addition, the prediction of plant parameters can be applied in early warning applications to assist operators in predicting future events. However, it is one of the most challenging tasks for operators because of the complex physical processes, nonlinear parameter variations, multiple system factors, and uncertainty of accidents [8].

To address these issues, several researchers have applied operator support systems and algorithms to reduce the burden on operators. For instance, Hsieh et al., used an abnormal symptom matrix [9], while Kim and Jung,. used a flowchart of the AOPs generated through the R software to implement operator support systems [10]. Other works using support vector machines (SVM) [11], expert systems [12], and artificial neural networks (ANNs) [13] have demonstrated the applicability of artificial intelligence (AI) techniques. However, the SVM algorithms and expert systems have their downsides. SVM algorithms are not suitable for large datasets, while expert systems are usually developed for specific domains and acquiring the necessary knowledge is time-consuming. Meanwhile, ANNs are regarded as one of the most relevant approaches to handle pattern recognition and large nonlinear data (e.g., handwriting recognition, translation, financial forecasting). Therefore, some papers have proposed algorithms using ANNs for the diagnostic and prognostic in NPPs [14-18].

Several diagnostic algorithms using ANNs have performed well in trained cases, but some drawbacks are observed. It cannot correctly assess anonymous cases as an unknown situation if an unknown abnormal situation is provided. As some abnormal events are not known and are unpredictable in actual NPPs, defining all abnormal events is also difficult. This limitation could harm the safety of NPPs caused by the wrong diagnostic results from an algorithm. In addition, the diagnostic algorithms need potential improvements. An improvement with this diagnostic algorithm is that the algorithm produces the diagnosis result with probabilities and the operators need to make the final decision based on the probability result. In some situations, this diagnostic algorithm may provide multiple and competing diagnosis results with competing probability values. For example, this diagnostic algorithm produces an output such that Event-A has a probability of 0.6 and Event-B has a probability of 0.4. In this case, operators might make errors due to confusion and uncertainty when the probabilities of diagnosis results are competing, especially during the initial period of abnormal situations when the probabilities of those events may not be clearly distinguished. In some abnormal situations, fast decision making is important. Hence, confirming the diagnostic result of the algorithm during the initial period of an abnormal situation is necessary.

In addition, many prediction algorithms using ANNs have shown good performances in single-step prediction, but have limitations. However, these studies can only predict the future single-step, which is limited to practical monitoring and early warning applications. The single-step prediction does not significantly benefit real early warning applications because the event after a multi-step condition is difficult to predict. Hence, multi-step prediction is more suitable for long-term prediction than single-step prediction; however, it is difficult to perform because of the lack of information and uncertainty or error accumulation [19]. Further, in the prediction using ANNs, the uncertainty is inevitable and the information about the uncertainty should be provided for the operators if the algorithm is supposed to support operator's prediction.

Therefore, this study not only aims to propose a diagnostic algorithm for abnormal situations by identifying unknown events, confirming the diagnostic results, and conducting the final diagnosis through the algorithm but also a long-term prediction algorithm with uncertainty estimation. The diagnosis algoritm combines long short-term memory (LSTM) and variational autoencoder (VAE) for identifying unknown situations and confirms the diagnosis of the

LSTM network. While, prediction algorithm aims to predict the long-term behavior of plant parameters for 2 hours with 120 steps as well as to provide uncertainty estimation using bidirectional LSTM (Bi-LSTM), attention mechanism (AM), and Variational encoder-decoer (VED). The proposed algorithms were trained and implemented using a compact nuclear simulator (CNS) in which the reference plant was a three-loop Westinghouse 900 MWe pressurized water reactor. Moreover, Gaussian noise was incorporated to all test input data to mimic real NPPs because the CNS produces data without noise.

Ⅱ. Methods

A. Long Short - Term Memory

LSTM is a neural network architecture based on the recurrent neural network (RNN) for processing long temporal sequences of data. LSTM has been suggested for long temporal sequence learning to deal with the vanishing gradient problem. Although LSTM has the same structure as RNNs, it uses a different equation to calculate the hidden state. Fig.1 shows the architecture of the LSTM cell applied in this study. The input sample x passes through every gate as in a conveyor belt system. With other LSTM models, each LSTM cell adjusts the output value using the input gate, forget gate, and output gate while maintaining the cell state. The information in the cell state is unchanged and can be added or deleted through each gate. Furthermore, as the operation of each gate is composed of additional operations attached to the cell state, it can avoid the vanishing gradient problem [14, 15, 19, 20].

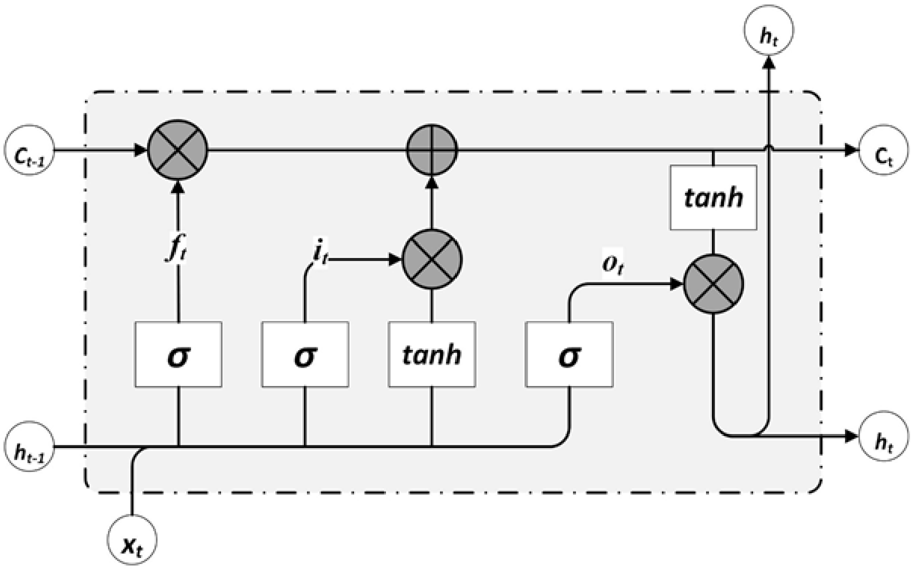


Fig. 1. Architecture of LSTM.

The input gate determines the capacity of the input value, the forget gate determines the degree to which the previous cell state is forgotten, and the output gate determines the value of the output. Eqs. (1)-(3) represent the input gate, forget gate, and output gate denoted by i, f, and o, respectively; Here, σ and t represent a sigmoid function and time state, respectively. The cell state is determined as shown in Eq. (4), where C represents the cell state. Finally, LSTM provides the output using Eq. (5), where h represents the output of the LSTM network [19].

$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f) \tag{1}$$

$$i_t = \sigma(W_i \cdot [C_{t-1},\ h_{t-1}, x_t] + b_i) \tag{2}$$

$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o) \tag{3}$$

$$C_t = f_t * C_{t-1} + i_t * \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{4}$$

$$h_t = o_t * \tanh(C_t) \tag{5}$$

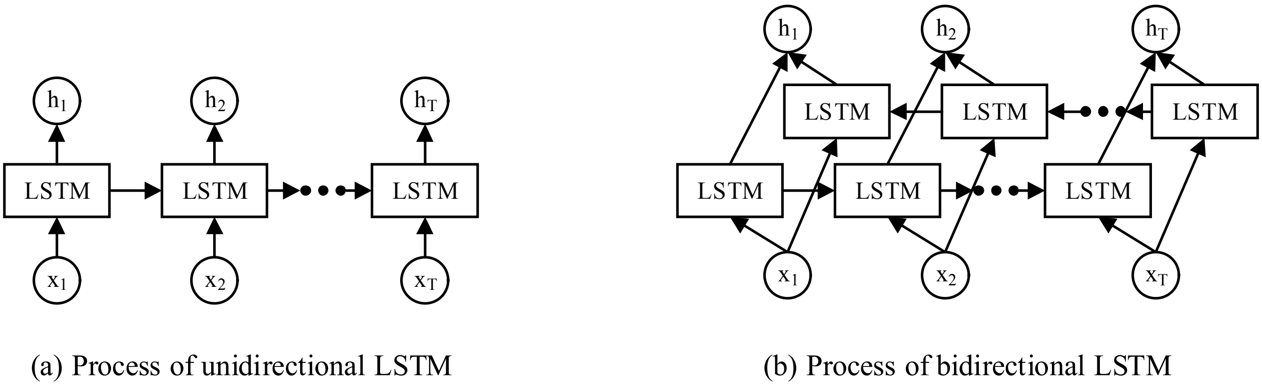
B. Bidirectional Long Short - Term Memory

However, LSTM only makes use of the forward dependencies. Long-term dependencies of LSTM are trained from chronologically sorted input data by only considering forward dependencies; whereas backward dependencies trained from reverse time-ordered data were not considered. As regards dependence on the prediction problem, all information contained in the time-series data should be fully used. But in the unidirectional process, it is highly possible that useful information is filtered out or not efficiently passed through the chain-like gated structure. Therefore, it could be informative to consider backward dependencies, which pass information in a reverse direction, into consideration [21]. To address this issue, Bi-LSTM with the ability to deal with both forward and backward dependencies is adopted in this study.

The Bi-LSTM is an extension of the described LSTM models in which two LSTMs are applied to the input data. Fig. 2 shows the process of unidirectional LSTM and bidirectional LSTM. The prediction results are influenced by not only the initial inputs but also the subsequent inputs in some regression problems. Bi-LSTM can improve the prediction accuracy by integrating the initial and subsequent inputs. Bi-LSTM is divided into forward LSTM and backward LSTM. The final output of Bi-LSTM is determined by the results of both forward and backward calculations, whose structures are consistent with the structure of the hidden state of LSTM: one direction process uses a forward hidden state from $t=1$ to T; the other direction process uses a backward hidden state from $t=T$ to 1. The output of the t^{th} Bi-LSTM is shown in Eq. (6) [21-23].

$$h_t^{BiLSTM} = \left[\overrightarrow{h}_t \oplus \overleftarrow{h}_t\right] \tag{6}$$

Where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ denote forward hidden state and backward hidden state, respectively. Here, it uses an element-wise sum to combine the forward and backward pass outputs.



(a) Process of unidirectional LSTM

(b) Process of bidirectional LSTM

Fig. 2. Process of unidirectional LSTM and bidirectional LSTM.

C. Softmax

The softmax function is used for the post-processing of the LSTM output. The softmax function is an activation function commonly used in the output layer of deep learning models that can categorize more than three classes of output [24]. Softmax significantly deviations between the values and then normalizes the outputs. Eq. (7) represents the softmax function. For $y \in R^k$ (the input vector to the softmax function), k is the number of classes of output. For $S(y)_1 \cdots S(y)_k$, the normalized output values were between 0 and 1, and the sum of output values was always 1.

$$S(y)_i = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}} \quad (\text{for } i = 1, \ \ldots, \ k) \tag{7}$$

D. Sequence to Sequence Learning

Seq2Seq learning was first successfully applied to real natural language processing by Sutskever et al [27]. It is a MIMO framework that aims to depict the inner regularities between highly structured inputs and outputs. This framework has been applied to machine translation and is effective in capturing continuous spatial and temporal representations of input sequences.

The architecture of the Seq2Seq network is shown in Fig. 3. It comprises two sets of LSTM blocks that serve as the encoder and decoder. The encoder processes the input sequence $x_1,\dots,x_T$ of length T and yields a summary of the past input sequence via the last hidden state h_T. After T times of recursive updates from Eqs. (1) - (5), the encoder summarizes the entire input sequence into a fixed-length vector. Subsequently, the encoder passes h_T to the decoder such that it is used as the input for sequence generation. The decoder recursively generates the output sequence $y_1,\dots,y_N$ of length N. Mathematically, the goal of the encoder and decoder LSTM is to estimate the conditional probability $p(y_1,\dots,y_N|x_1,\dots,x_T)$. The encoder and decoder compute this conditional probability by first obtaining the fixed-dimensional representation of the input sequence provided by the last hidden state of the encoder LSTM. Subsequently, the probability of the output sequence is computed as shown in Eq. (8) [25].

$$p(y_1,\dots,y_N|x_1,\dots,x_T)=\prod_{n=1}^{N}p(y_n|y_1,\dots\ y_{n-1},\ context) \tag{8}$$

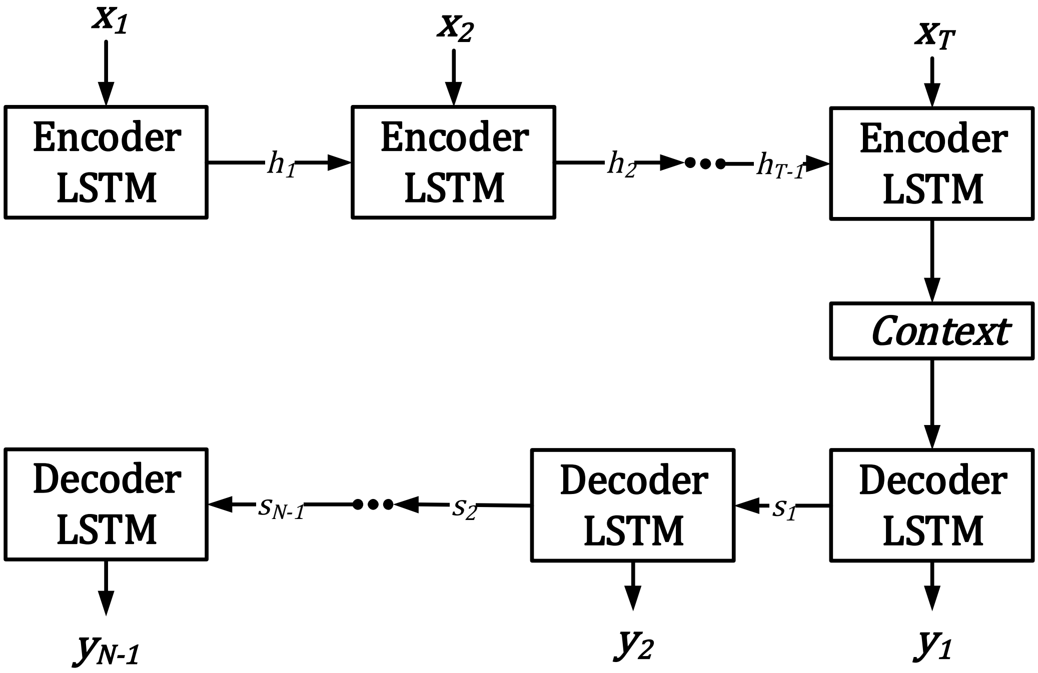


Fig. 3. The framework of encoder-decoder LSTM network.

E. Attention Mechanism

Standard Seq2Seq learning must be able to compress all the necessary information of the source data into a fixed-length vector. In this case, information loss is inevitable, and the prediction effect will become increasingly worse with the continuous growth of the time-series sequence [26]. This can render it difficult for the network to cope with long sequences, particularly those that are longer than the sequences in the training input data [27].

Hence, the AM has garnered significant interest in deep learning. It is a probability-weighting mechanism that mimics the attention of the human brain. When the human brain observes objects, it focuses on specific locations and disregards other locations. The AM increases the accuracy of the model by highlighting more important factors by assigning different probability weights. It has been successfully applied to machine translation, video analysis, and other related fields.

Applying the AM to the encoder - decoder LSTM network allows the neural network to adaptively focus on input features that are more important to the current output and mitigate the interference of other features. Fig. 4 shows an architecture of AM. Using the encoder hidden state states $H=[h_1,h_2,\ldots,h_T]$ as the input of the AM, the AM will search for the attention weight of the hidden state s_j of the LSTM decoder, where the attention weight α_{ji} between the output at time j and the input at time i is calculated using Eqs. (9) and (10). The context vector $Context=[context_1,context_2,\ldots,context_T]$ can be obtained by multiplying the attention weight α_{ji} and the encoder hidden states H, as shown in Eq. (11). [26, 28].

$$e_j = H \bullet s_j \tag{9}$$

$$\alpha_{ji} = \frac{\exp(e_{ji})}{\sum_{k=1}^{T} \exp(e_{jk})} \tag{10}$$

$$context_j = \alpha_{ji} \bullet H \tag{11}$$

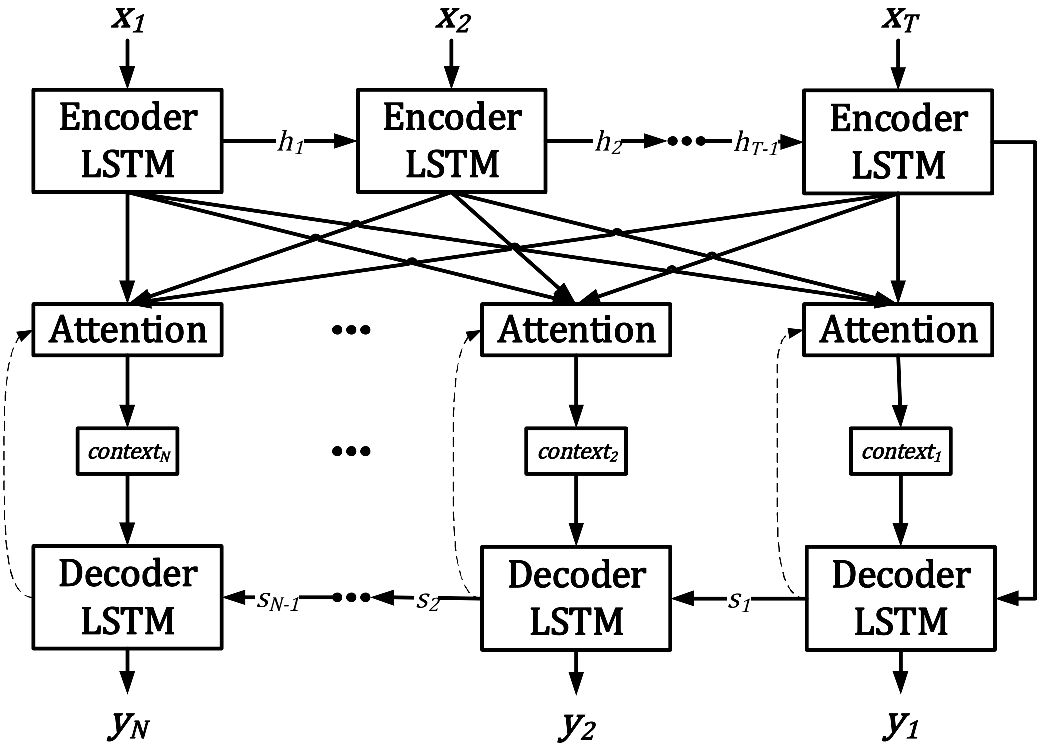


Fig. 4 Architecture of AM

F. Variational Autoencoder

VAE [29] is an unsupervised deep learning generative model that can model the distribution of training data. If input data are similar to the training data, the output appears to be similar to the input. Otherwise, a probabilistic measure that considers the variability of the distribution variables decreases. Several studies have suggested a fault detection algorithm using the reconstruction log-likelihood of VAE and showed the compatibility of VAE with LSTM [30-32].

VAE provides a flexible formulation for interpreting and encoding z as a potential variable in probabilistic generation models. As shown in Fig. 5, the input sample x passes through the encoder to obtain parameters of latent space distribution. The latent variable z was obtained from sampling in the current distribution, and then z was used to generate a reconstructed sample through the decoder (Chen et al., 2019). VAE comprises of a probabilistic encoder ($q_{\phi}(z|x)$) and decoder ($p_{\theta}(x|z)$). As the posterior distribution ($p_{\theta}(z|x)$) is intractable, VAE approximates $p_{\theta}(z|x)$ using the encoder $q_{\phi}(z|x)$, which is assumed to be Gaussian and is parameterized by $\varnothing$. This enables the encoder to learn and predict latent variables z, which makes it is possible to draw samples from this distribution.

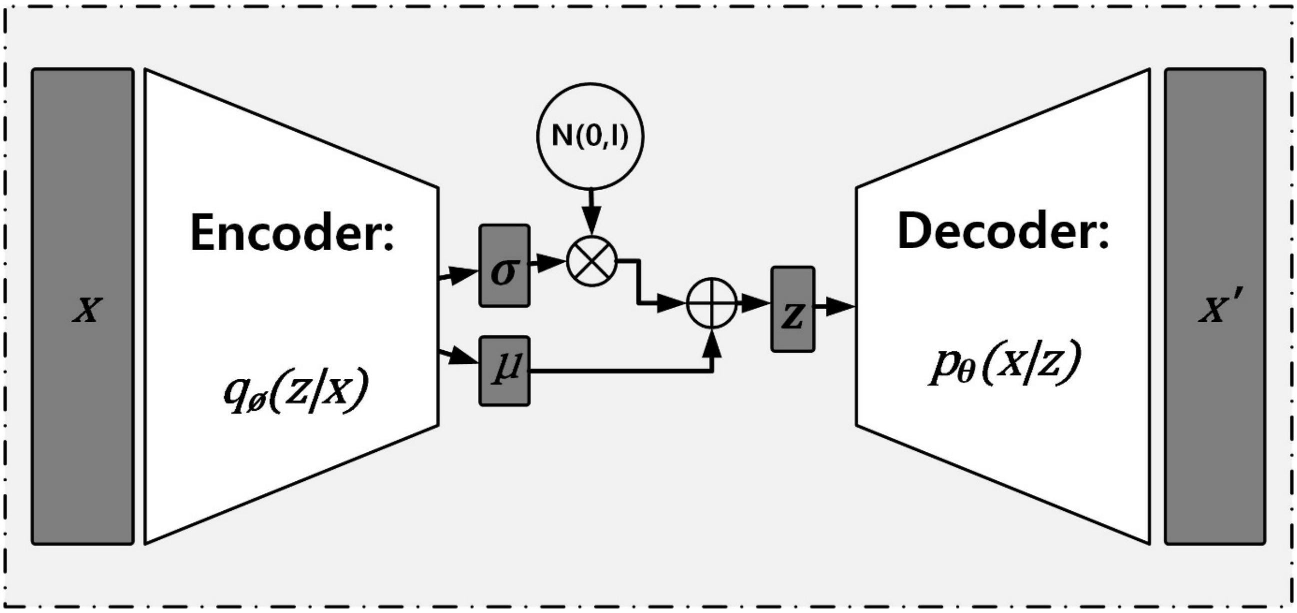


Fig. 5. Architecture of VAE.

To decode a sample z drawn from $q_\phi(z|x)$ to the input x, the reconstruction loss (Eq. (12)) must be minimized. The first term of Eq. (12) is the Kullback-Leibler (KL) divergence between the approximate posterior and prior latent variable . This term makes the posterior distribution to be similar to the prior distribution by working as a regularization term. The second term of Eq. (12) can be understood in terms of reconstruction of x through the posterior distribution $q_\phi(z|x)$ and likelihood $p_\theta(x|z)$.

$$L(\theta,\phi;x) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + E_{q_\phi(z|x)}[\log p_\theta(x|z)] \qquad (12)$$

The choice of distribution types is important because VAE models the approximated posterior distribution $q_\phi(z|x)$ from a prior $p_\theta(x)$ and likelihood $p_\theta(x|z)$. A typical choice for the posterior is Gaussian distribution, where the standard normal distribution $N(0,1)$ is used for the prior $p_\theta(x)$.

G. Variational Encoder-Decoder

In some applications, the input data are transformed into output data (e.g., machine translation, multi-step prediction, and natural language processing). In these tasks, a VAE is insufficient, and an encoder‐decoder framework is required. Different efforts have been expended to extend the VAE to encoder‐decoder frameworks, which transform an input x to an output y. We denote our model distribution as $p(y|x,z)$. We introduce a latent variable z with a standard Gaussian prior and factor $p(y|x,z)$, as shown in Eq. (13).

$$p(y|x)=\int_z p(y|z,x)p(z|x)dz \tag{13}$$

For the definition expressed in Eqs. (12) and (13), the variational lower bound of the VED can be formulated as shown in Eq. (14).

$$L_{VED}(\theta,\phi;x,y)=-D_{KL}(q_{\phi}(z|x,y)||p_{\theta}(z|x))+E_{q_{\phi}(z|x,y)}[\log p_{\theta}(y|z,x)] \tag{14}$$

where $p_{\theta}(z|x)$ is the prior model of the VED, $q_{\phi}(z|x,y)$ is the posterior approximator of the VED, and $p_{\theta}(y|z,x)$ is the decoder with guidance from z [33].

Ⅲ. Development of Diagnostic Algorithm for Abnormal Situations

A. Diagnostic Algorithm Design

This section describes the overall structure of the diagnostic algorithm for abnormal situations using ANN. Fig. 6 illustrates the functional architecture of the diagnostic algorithm design, which consists of four functions: 1) pre-processing function, 2) unknown event identification function, 3) event diagnosis function, and 4) confirmation of diagnosis result function.

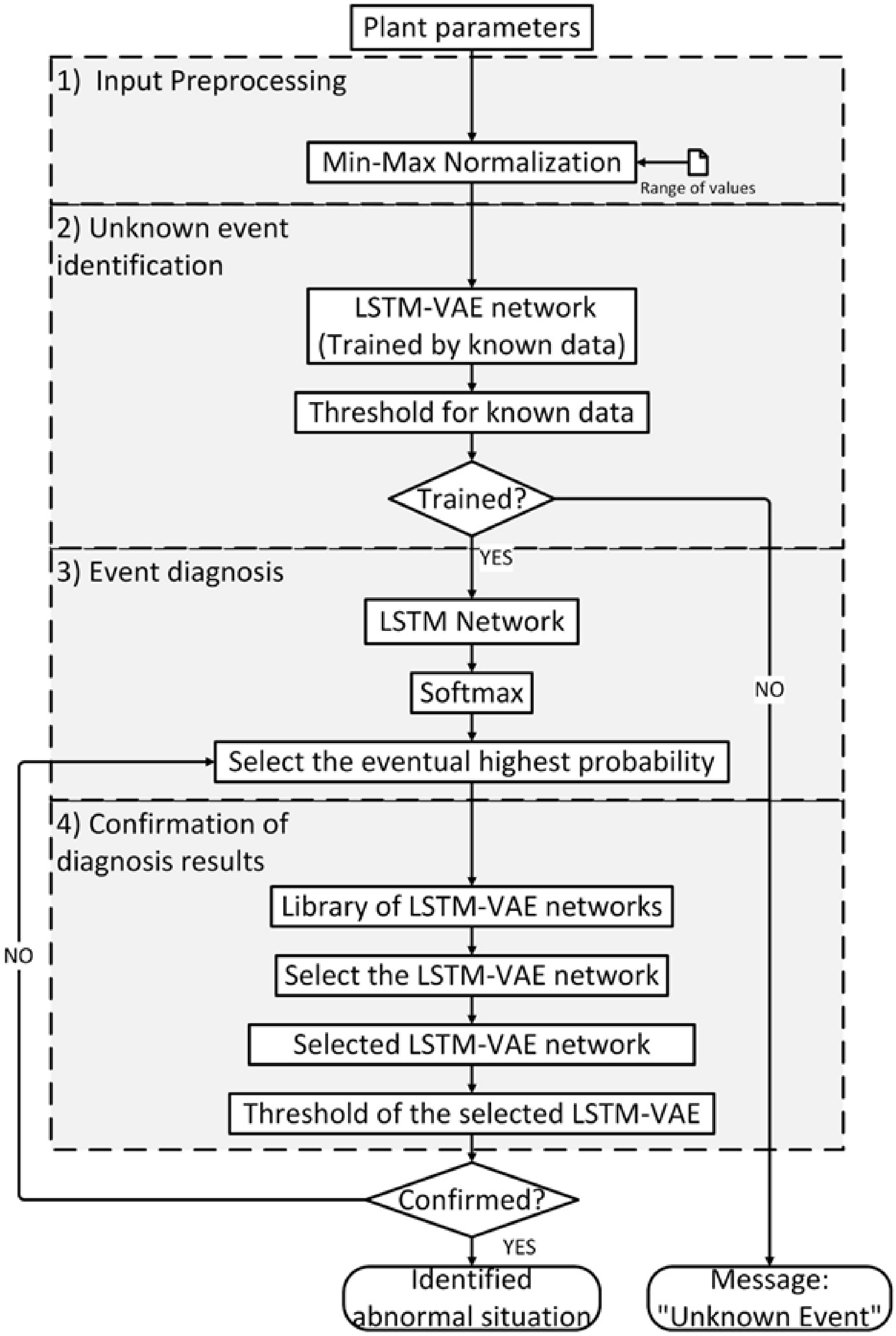

Plant parameters
1) Input Preprocessing
Min-Max Normalization
Range of values
2) Unknown event identification
LSTM-VAE network (Trained by known data)
Threshold for known data
Trained?
YES
NO
3) Event diagnosis
LSTM Network
Softmax
Select the eventual highest probability
4) Confirmation of diagnosis results
Library of LSTM-VAE networks
Select the LSTM-VAE network
Selected LSTM-VAE network
Threshold of the selected LSTM-VAE
NO
Confirmed?
YES
Identified abnormal situation
Message: "Unknown Event"

Fig. 6. Functional architecture of the diagnostic algorithm design.

1. Pre-processing Function

The first function of the algorithm is to process the plant parameters and make it suitable as network inputs. The inputs for the networks are selected from operating procedures based on their importance and ability to affect the state of the plant or system availability. These inputs should have a range of values from 0 to 1. However, plant parameters have a different range of values or states (e.g., pressurizer pressure: 158 kg/cm2, alarm: on or off). Generally, variables with higher values will have a larger impact on the network results. However, higher values are not necessarily more important for prediction. This problem produces local minima. Therefore, the input pre-processing obtains the regular plant parameters as input and then outputs the normalized plant parameters that will be utilized by the networks.

Min-max normalization is used to prevent local minima and increase the learning speed. Thus, the input of the networks is calculated by using Eq. (15). here, X is the current value of plant parameters, and $X_{\min}$ and $X_{\max}$ are the minimum and maximum values of collected data, respectively. In this equation, X_{input} has a range of 0 - 1.

$$X_{input} = \frac{(X - X_{\min})}{(X_{\max} - X_{\min})} \tag{15}$$

2. Unknown Event Identification Function

This function is used to identify the unknown event via a combination of the VAE and LSTM networks. The VAE network was combined with LSTM [31] and it was used not only to support the sequence of input data but also to capture the complex temporal dependence in time series. This function receives normalized NPP parameters from the input pre-processing function and identifies the unknown event in real time. The anomaly scores that indicate discrepancies between the actual and trained data were used for this function. If the anomaly score is below the threshold, the event is identified as a known event for which the diagnosis network in the next function has been trained. If the anomaly score is above the threshold, the event is unknown, and the message “Unknown event occurrence” is provided to the operators as the output.

The process of unknown event identification function is shown in Fig. 7. To consider the temporal dependency of time-series data in a VAE, a VAE is combined with LSTMs by replacing the feed-forward network in a VAE with the LSTMs similar to conventional temporal autoencoders. Given the multi-parameters input $x_t \in R^D$ (where D is the number of input parameters), made up of $x_{1,\ t}$, $\cdots$ $x_{D,t}$, at t time which is normalized plant parameters from input pre-processing function, the encoder approximates the posterior $p(z_t|x_t)$ by the LSTM of the encoder to estimate the mean $\mu_{z_t} \in R^M$ and variance $\sigma_{z_t} \in R^M$ of the latent variable $z_t \in R^M$. Then, the randomly sampled z_t from the posterior $p(z_t|x_t)$ is fed into the LSTM of the decoder. The final outputs are the reconstruction mean $\mu_{x_t} \in R^D$ $(\mu_{x_{1,t}},\ \ldots\ \mu_{x_{D,t}})$ and variance $\sigma_{x_t} \in R^D$ $(\sigma_{x_{1,t}}, \ldots\ \sigma_{x_{D,t}})$.

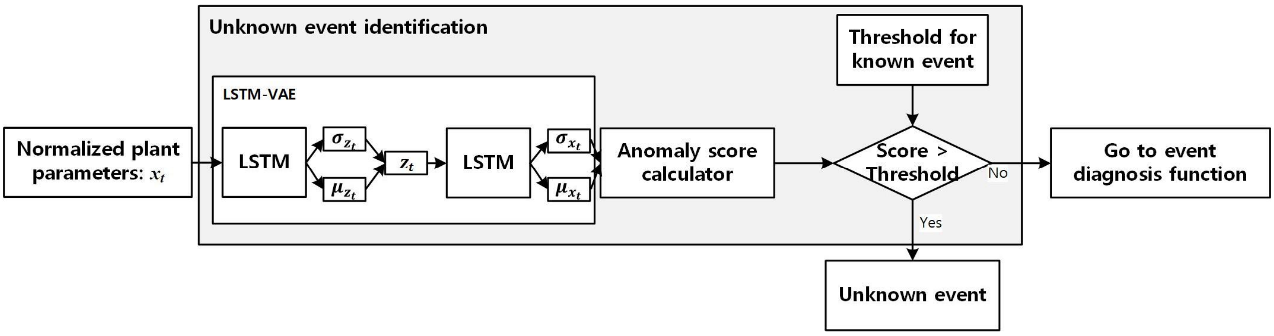


Fig. 7. Various processes in the unknown event identification function.

To identify the unknown event, the unknown event identification function detects an anomalous execution when the current anomaly score is above the threshold α' in Eq. (16). The term $f_s(\ x_t,\ \varnothing,\ \theta)$ is an anomaly score calculator. The anomaly score is defined as the negative log-likelihood of x_t, represented in Eq. (17), with respect to the reconstructed distribution of x_t from an LSTM-VAE network. Here, μ_{x_t} and σ_{x_t} are the mean and variance of the reconstructed distribution from an LSTM-VAE network with parameters $\varnothing$ and θ, respectively. Fig. 8 shows an example of the anomaly score calculation. Note that $x_t \in R^3$ (for $x_{1,t}$, $x_{2,t}$, $x_{3,t}$) represents the normalized input parameters, $\mu_{x_t} \in R^3$ (for $\mu_{x_{1,t}}$, $\mu_{x_{2,t}}$, $\mu_{x_{3t}}$) represents the reconstruction mean, and $\sigma_{x_t} \in R^3$ (for $\sigma_{x_{1,t}}$, $\sigma_{x_{2,t}}$, $\sigma_{x_{3t}}$) represents the reconstruction variance. Each element goes through the Eq. (17) then the anomaly score is calculated. Notice that in this example, the number of input parameters (D) is three. Therefore, the anomaly score calculated in this example is 0.927 as shown in Fig. 8. A high anomaly score means that the input has not been adequately reconstructed by the LSTM-VAE network.

$$\begin{cases} Unknown\ event, & \text{if } f_s(\ x_t,\ \varnothing,\ \theta)\rangle\ \alpha' \\ Known\ event, & otherwise, \end{cases} \tag{16}$$

$$f_s(x_t,\ \varnothing,\ \theta) = -\frac{1}{D}\sum_{i=1}^{D}\log p(x_{i,t};\mu_{x_{i,t}},\sigma_{x_{i,t}}) \tag{17}$$

$$\alpha' = S_{mean} + 3*S_{std} \tag{18}$$

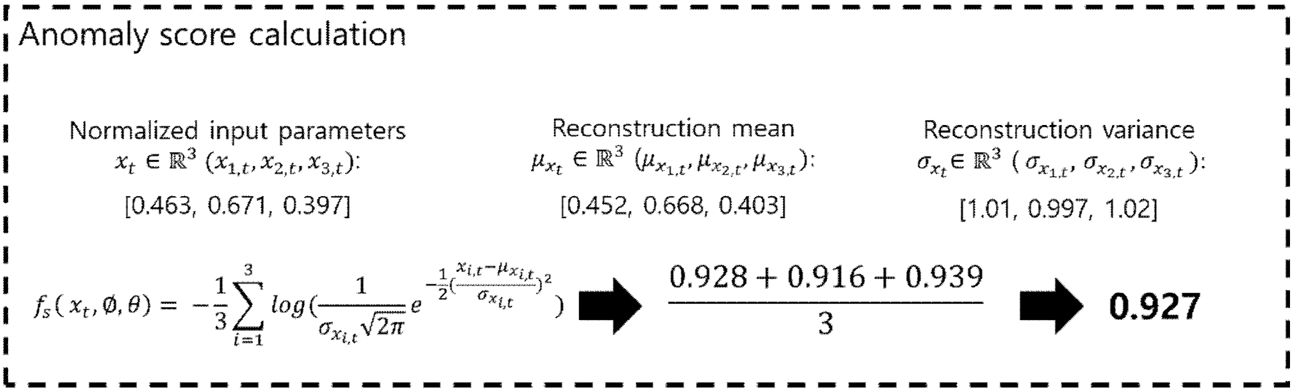

Anomaly score calculation
Normalized input parameters
$x_t \in \mathbb{R}^3\ (x_{1,t}, x_{2,t}, x_{3,t})$:
[0.463, 0.671, 0.397]
Reconstruction mean
$\mu_{x_t} \in \mathbb{R}^3\ (\mu_{x_{1,t}}, \mu_{x_{2,t}}, \mu_{x_{3,t}})$:
[0.452, 0.668, 0.403]
Reconstruction variance
$\sigma_{x_t} \in \mathbb{R}^3\ (\sigma_{x_{1,t}}, \sigma_{x_{2,t}}, \sigma_{x_{3,t}})$:
[1.01, 0.997, 1.02]
$f_s(x_t, \emptyset, \theta) = -\frac{1}{3}\sum_{i=1}^{3} log(\frac{1}{\sigma_{x_{i,t}}\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x_{i,t}-\mu_{x_{i,t}}}{\sigma_{x_{i,t}}})^2})$
$\frac{0.928 + 0.916 + 0.939}{3}$
0.927

Fig. 8. Example of anomaly score calculation.

The threshold α' was determined using three-sigma limits after considering the anomaly score distribution of the training data. The fact that the anomaly scores are achieving a smaller value indicates that the output data (i.e., reconstructed data from LSTM-VAE) is similar to the training data and the threshold considers the upper control limit. S_{mean} and S_{std} are the mean and standard deviation of the anomaly scores of the training data, respectively (Eq. (18)). This not only sets a range for the process parameter at 0.27 % control limits (corresponding to the three-sigma in normal distribution) but also minimizes the cost associated with preventing the error of classifying a known event as unknown. Section Ⅲ.B.2 discusses the method to determine the threshold and hyperparameters.

3. Event Diagnosis Function

This function produces diagnosis results of the plant situation using LSTM. Fig. 9 shows the processes in the event diagnosis function. The LSTM network receives normalized plant parameters from the input pre-processing function and produces identified abnormal events with their probabilities. The output is post-processed using the softmax function. The probability represents the confidence level of the identified event. Then, this function selects the event with the highest probability among the diagnostic results for the confirmation function. In addition, multiple events can be identified with different probabilities in the previous function. If the confirmation function returns the information stating that the current situation is not consistent with the diagnostic result, then this function will select the next event with the highest probability until the current situation is consistent with the diagnostic result. The procedure to determine hyperparameters, such as number of layers and nodes, is described in Section Ⅲ.B.3.

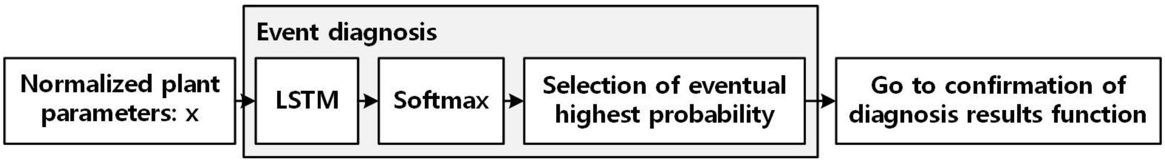

Event diagnosis
Normalized plant parameters: x
LSTM
Softmax
Selection of eventual highest probability
Go to confirmation of diagnosis results function

Fig. 9. Processes in the event diagnosis function.

4. Confirmation of Diagnosis Result Function

This function is used to confirm whether the current abnormal situation is identical to the event selected in the event diagnosis function. This function has a library that consists of LSTM-VAE networks for trained events. Further, the LSTM-VAE network for the selected event is used to confirm that the selected event is identical to the trained event.

Fig. 10 shows the confirmation process of the selected event. First, this function selects the LSTM-VAE network from the library that corresponds to the event identified in the previous function. Next, it verifies whether the current situation is identical to the selected event by using the LSTM-VAE network. To estimate the anomaly score, this function uses negative log-likelihood (i.e., similar to “the unknown event identification function”). If the negative log-likelihood is below the threshold, then the algorithm declares that the diagnosis result from the event diagnosis function is correct and confirmed. If the negative log-likelihood is beyond the threshold, then it returns to the previous function to select another event. The thresholds of LSTM-VAE networks are determined in a similar manner as that described in Section Ⅲ.A.2.

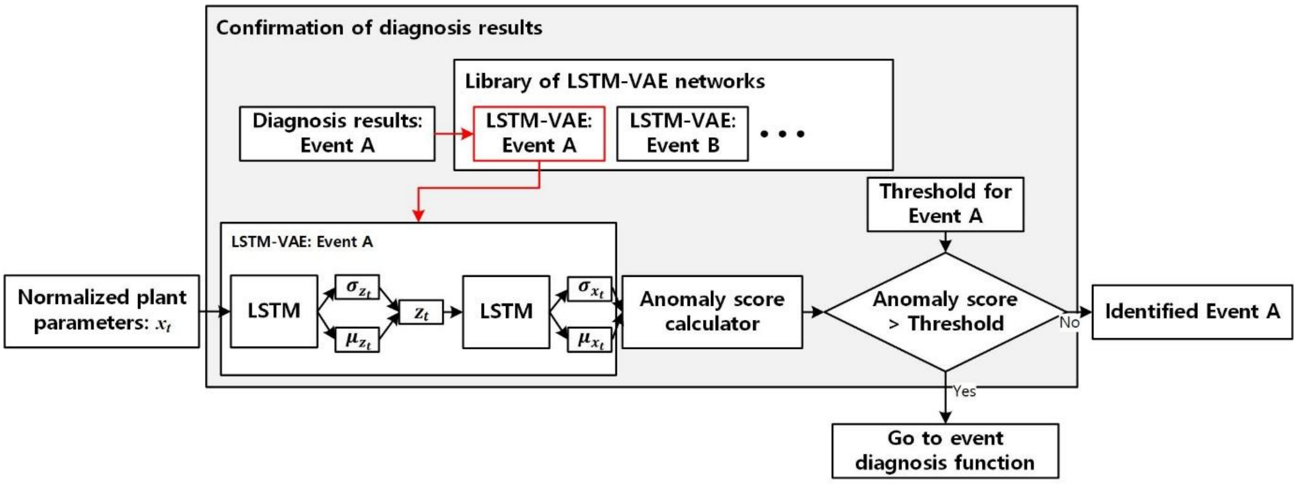


Fig. 10. Processes in the confirmation of diagnosis results function.

B. Experiment

The suggested algorithm was implemented by using a CNS that simulates a Westinghouse 900 MWe, three loops, pressurized water reactor. Fig. 11 shows a plant overview interface of the CNS. For implementation, a desktop computer with NVIDIA GeForce GTX 1080 11 GB GPU, Intel 4.00 GHz CPU, Samsung 850 PRO 512 GB MZ-7KE512B SSD, and 24 GB RAM was used. Python 3.7.3 was used as the coding language, and several python libraries, including Keras and Pandas, were used to model the algorithm.

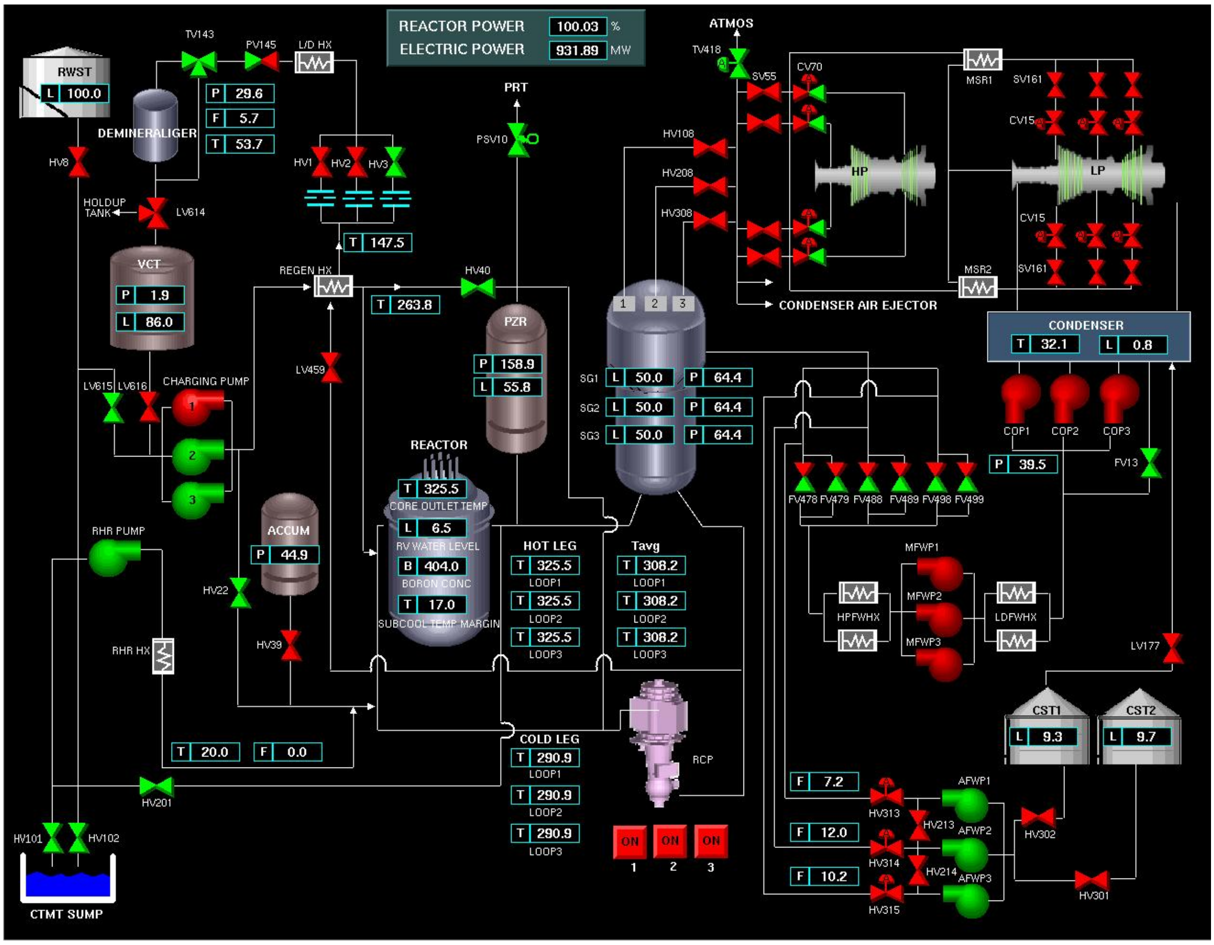


Fig. 11. Plant overview interface of CNS.

1. Data Collection

To implement, train, and validate the algorithm, CNS was used as a real-time testbed. A total of 20 abnormal situations and 558 cases were simulated to collect data. Table 2 shows the abnormal scenarios and the numbers of simulations for each event. The scenarios included representative abnormal situations in actual NPPs, such as instrument failures (Nos. 1 – 6), component failures (Nos. 7 – 16), and leakages (Nos. 17 – 20).

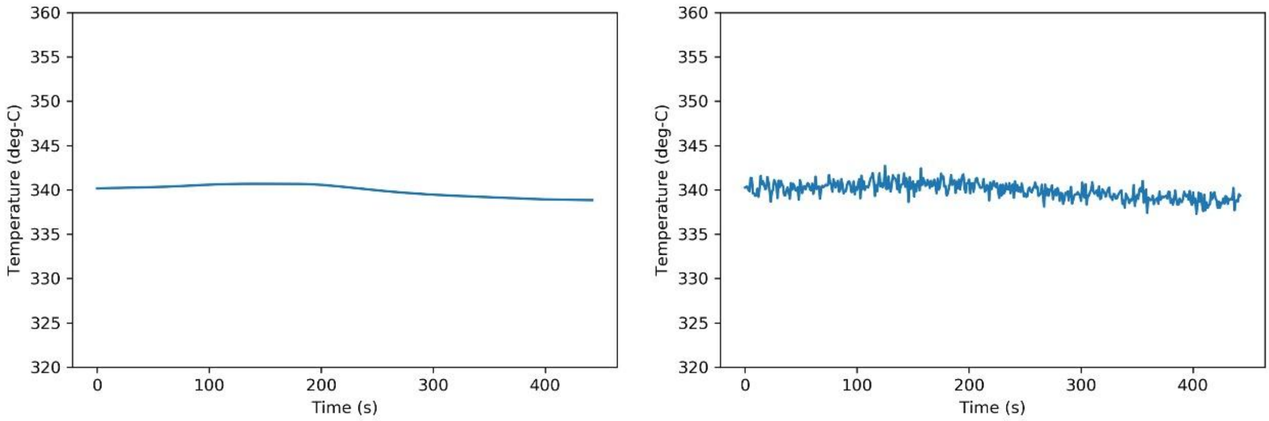
Table 1. Abnormal scenarios and the number of simulations.

No.	Scenarios	Training Cases	Verification Cases	Total Cases
1	Failure of pressurizer pressure channel (High)	14	4	18
2	Failure of pressurizer pressure channel (Low)	20	6	26
3	Failure of pressurizer water level channel (High)	-	6	6
4	Failure of pressurizer water level channel (Low)	11	4	15
5	Failure of steam generator water level channel (Low)	32	8	40
6	Failure of steam generator water level channel (High)	35	6	41
7	Control rod drop	38	10	48
8	Continuous insertion of control rod	7	1	8
9	Continuous withdrawal of control rod	6	2	8
10	Opening of pressurizer power-operated relief valve	42	10	52
11	Failure of pressurizer safety valve	35	8	43
12	Opening of pressurizer spray valve	41	9	50
13	Stopping of charging pump	-	1	1
14	Stopping of two main feedwater pumps	-	3	3
15	Main steam line isolation	-	3	3
16	Rupture at the inlet of the regenerative heat exchanger	40	10	50
17	Leakage from chemical volume and control system to component coolant water (CCW)	40	10	50
18	Leakage at the outlet of charging control flow valve	24	6	30
19	Leakage into the CCW system from the reactor coolant system	24	6	30
20	Leakage from steam generator tube	-	36	36
	Total	409	149	558

Based on the abnormal operating procedures of the reference plant, 139 parameters were selected for input, including plant variables (e.g., temperature or pressure) and component states (e.g., pump or valve status). These parameters were collected every second during the simulations.

Among the 20 scenarios collected, 15 scenarios containing 409 cases were used for training, and 5 scenarios were used for validating untrained events. A total of 149 cases were used for validating the algorithm, including the five untrained events (i.e., 3, 13, 14, 15, and 20). Among them, 115 cases were used for determining the thresholds of identification of unknown event function and confirmation of diagnosis result function.

The collected data were added with ±5 % Gaussian noise to reflect actual signals from NPPs. The CNS produces data without noises, as shown in Fig. 12 (a). The noise was added to the CNS data intentionally, as shown in Fig. 12 (b).



(a) Pressurizer temperature (original CNS data)

(b) Pressurizer temperature (CNS data with Gaussian noise)

Fig. 12. Examples of pressurizer temperature data ((a): original CNS data and (b): data with ±5 % Gaussian noise).

2. Function Implementation

a. Pre-processing function

To implement the min-max normalization in the input preprocessing function, we determined the maximum and minimum values of parameters based on all the collected data (i.e., 558 cases). Moreover, as ±5 % Gaussian noise was added to the simulator data (making it similar to actual NPP data), some data can be larger than the maximum values or lower than the minimum values. Therefore, the data will not fall between 0 and 1 when normalized. To prevent this problem, we added a 10 % margin to the maximum and −10 % to the minimum values for each parameter. Fig. 13 shows an example of a normalized pressurizer temperature.

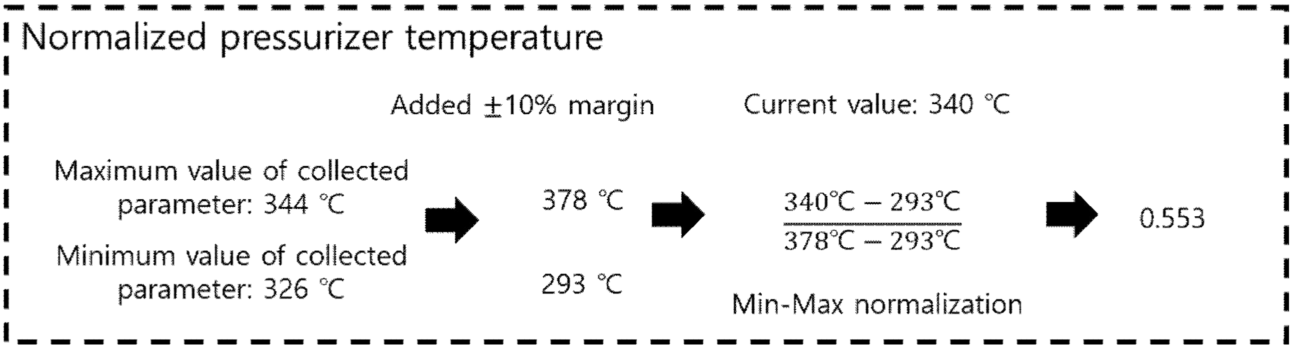

Normalized pressurizer temperature
Added ±10% margin
Current value: 340 ℃
Maximum value of collected parameter: 344 ℃
Minimum value of collected parameter: 326 ℃
378 ℃
293 ℃
$\frac{340℃ - 293℃}{378℃ - 293℃}$
0.553
Min-Max normalization

Fig. 13. Example of normalized pressurizer temperature.

b. Identification of unknown event function

The unknown event function was identified using the LSTM-VAE network. In this network, a dataset has 10 s sequence and 139 input values. As mentioned in Section 4.1, 409 scenarios (i.e., 192,637 datasets for 139 parameters) were trained. Fig. 14 illustrates how the unknown event function is identified by using the LSTM-VAE network. VAE does not necessarily tune the hyperparameters as it provides a variational information bottleneck that prevents overfitting. Consequently, it has the effect of making the optimal bottleneck size in a given hyperparameter [34-36]. Additionally, this LSTM-VAE network used Adam [37] optimizer with a learning rate of 0.0001 and ran for 100 epochs.

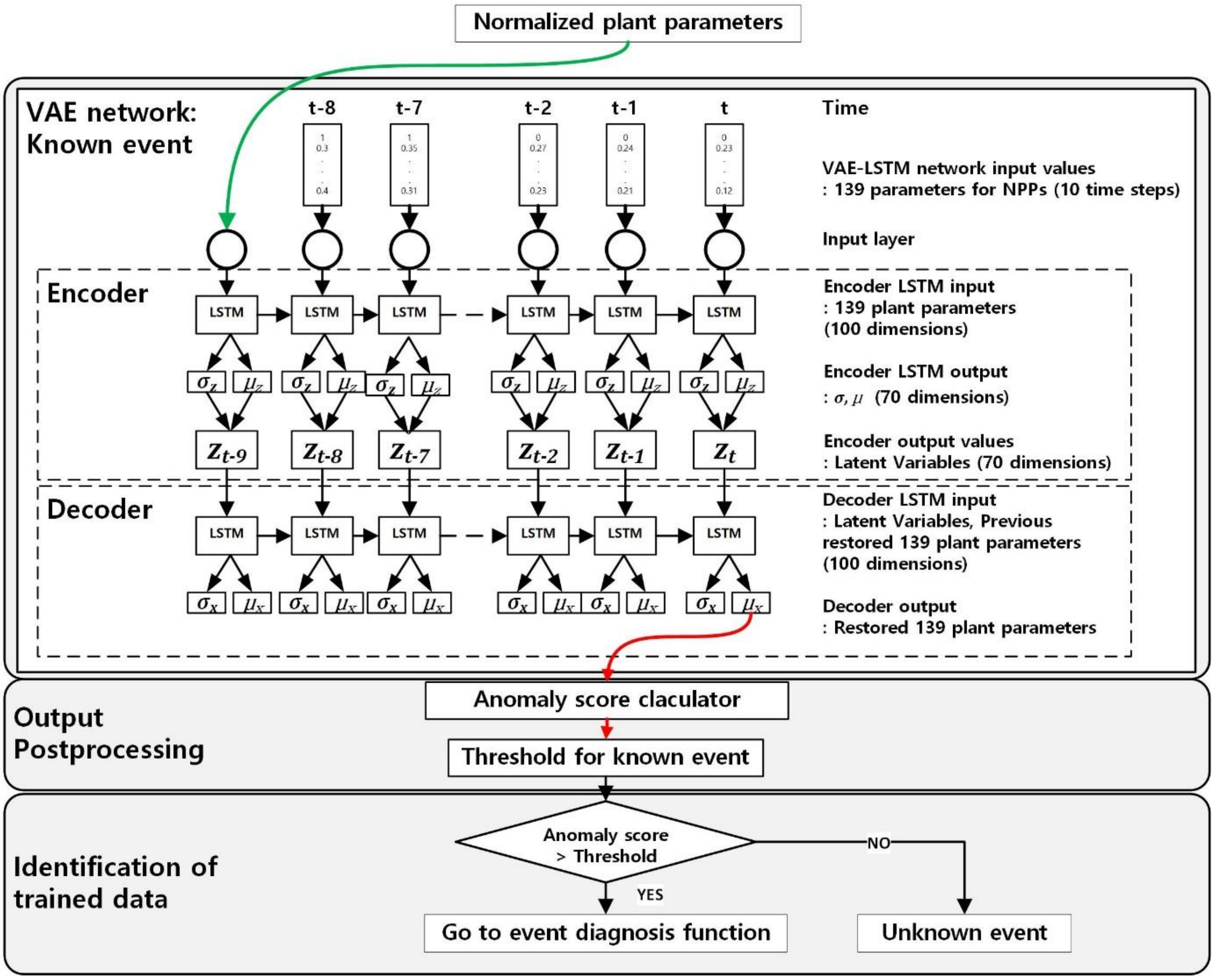


Fig. 14. Identification of unknown event function using the LSTM-VAE network.

This study used the receiver operating characteristic (ROC) curve to evaluate the performance of the network. The ROC curve is created by plotting the true positive rate (i.e., the ratio of correctly predicted positive observations to all observations in the actual class) against the false positive rate (i.e., the ratio of the incorrectly predicted negative observations to all observations in the actual class) is a useful method of interpreting the performance of a binary classifier. The area under the ROC curve (AUC) is an effective measure to summarize the overall diagnostic accuracy of the test and is interpreted as the average value of sensitivity for all possible values of specificity. It can also take on any value between 0 and 1, where a value of 0 indicates a perfectly inaccurate test. The closer AUC is to 1, the better the overall diagnostic performance. In general, an AUC of 0.5 suggests no discrimination, 0.7 to 0.8 is considered acceptable, 0.8 to 0.9 is considered excellent, and more than 0.9 is considered outstanding [38, 39]. The threshold was determined as 0.923 using Eq. (18). Fig. 15 shows the result of ROC and AUC of LSTM-VAE.

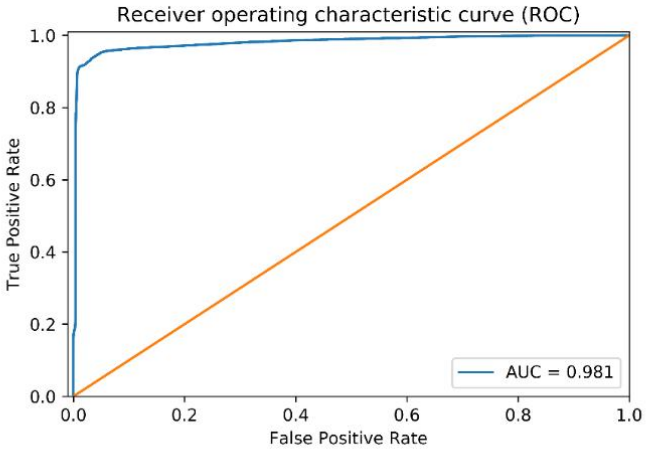


Fig. 15. Result of ROC and AUC of the identification of unknown event function.

c. Event diagnosis function

The event diagnosis function was developed using LSTM. A total of 409 scenarios (i.e., 192,637 datasets for 139 parameters) were trained. To optimize the LSTM network in the event diagnosis function, a manual search method was used while individually adjusting the hyperparameters (e.g., input sequence length, batch size, and number of layers). In general, there is no golden rule for hyperparameter determination to optimize the network [36, 39, 40]. Table 2 shows the accuracy comparison results of different configured networks. The accuracy is defined as the ratio of correctly predicted data to total verification data. Consequently, an optimal LSTM network with 10 s time steps, 32 batch sizes, and 2 layers was selected. This network used the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0001 and ran for 100 epochs. Fig. 16 shows the illustration of the event diagnosis function using LSTM and softmax.

Table 2. Accuracy comparison results of various configured networks.

No.	Time step	Batch size	Layers	Accuracy
1	5	32	2	0.9668
2	5	32	3	0.9638
3	5	64	2	0.9634
4	5	64	3	0.9650
5	10	32	2	0.9768
6	10	32	3	0.9746
7	10	64	2	0.9764
8	10	64	3	0.9741
9	15	32	2	0.9767
10	15	32	3	0.9762
11	15	64	2	0.9764
12	15	64	3	0.9766

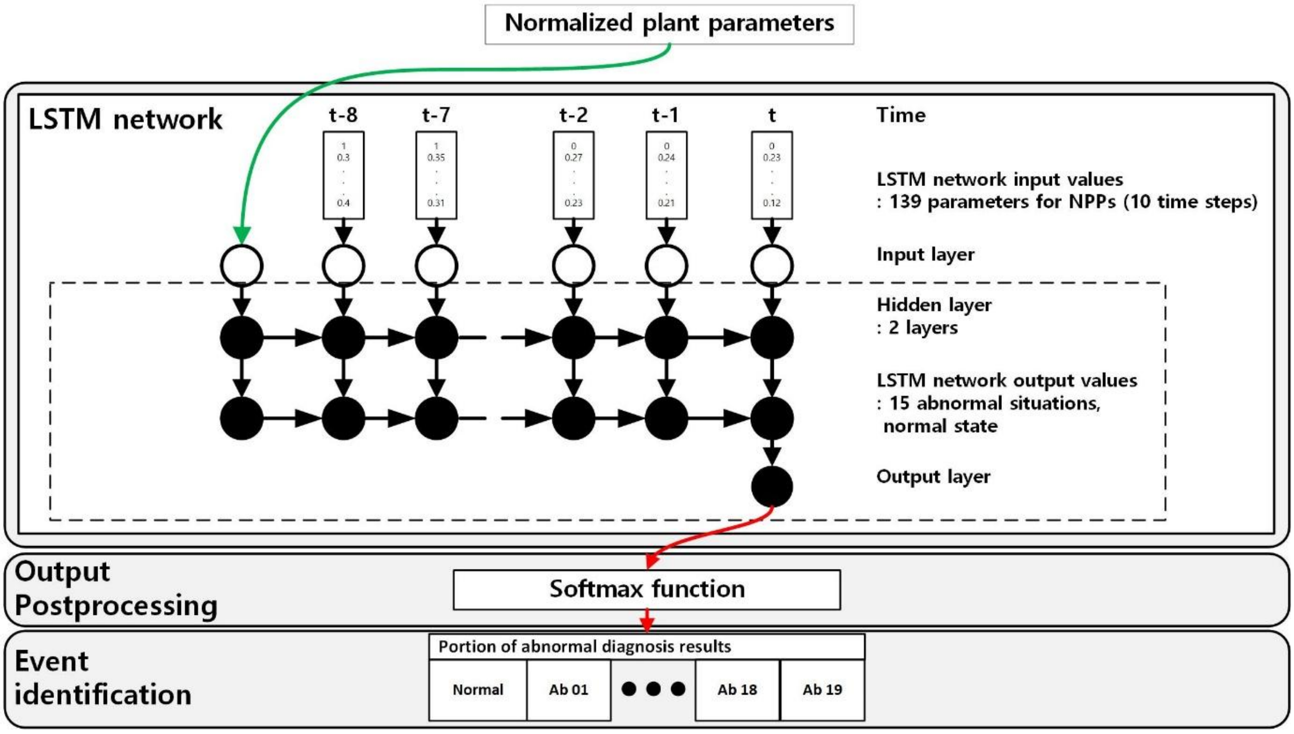


Fig. 16. Event diagnosis function using LSTM and softmax.

d. Confirmation of diagnosis result function

To implement this function, an LSTM-VAE library was developed. This library comprises 16 LSTM-VAE networks that are trained for each known event (i.e., 15 abnormal events and 1 normal state). Fig. 17 shows the illustration of the confirmation of diagnosis result function (i.e., when Ab 01 event is diagnosed). Each LSTM-VAE network and threshold were comprised and determined in a manner similar to the identification of an unknown event. 545 3 shows the training dataset, AUC, and threshold of each LSTM-VAE network in the confirmation of diagnosis result function.

Table 3. Results of training data, performance, and thresholds of LSTM-VAE networks.

No.	Name	Trained data	AUC	Threshold
1	Ab 01	3,459 datasets (Ab 01)	0.995	0.923
2	Ab 02	3,785 datasets (Ab 02)	0.934	0.923
3	Ab 04	5,658 datasets (Ab 04)	0.993	0.922
4	Ab 05	5,730 datasets (Ab 05)	0.998	0.921
5	Ab 06	8,312 datasets (Ab 06)	0.974	0.923
6	Ab 07	34,735 datasets (Ab 07)	0.999	0.920
7	Ab 08	2,831 datasets (Ab 08)	0.958	0.924
8	Ab 09	2,070 datasets (Ab 09)	0.955	0.927
9	Ab 10	8,394 datasets (Ab 10)	0.985	0.921
10	Ab 11	8,004 datasets (Ab 11)	0.938	0.921
11	Ab 12	23,885 datasets (Ab 12)	0.996	0.920
12	Ab 16	24,600 datasets (Ab 16)	0.965	0.920
13	Ab 17	30,904 datasets (Ab 17)	0.953	0.934
14	Ab 18	14,805 datasets (Ab 18)	0.984	0.922
15	Ab 19	1,546 datasets (Ab 19)	0.967	0.920
16	Normal	7,602 datasets (Normal state)	0.995	0.923

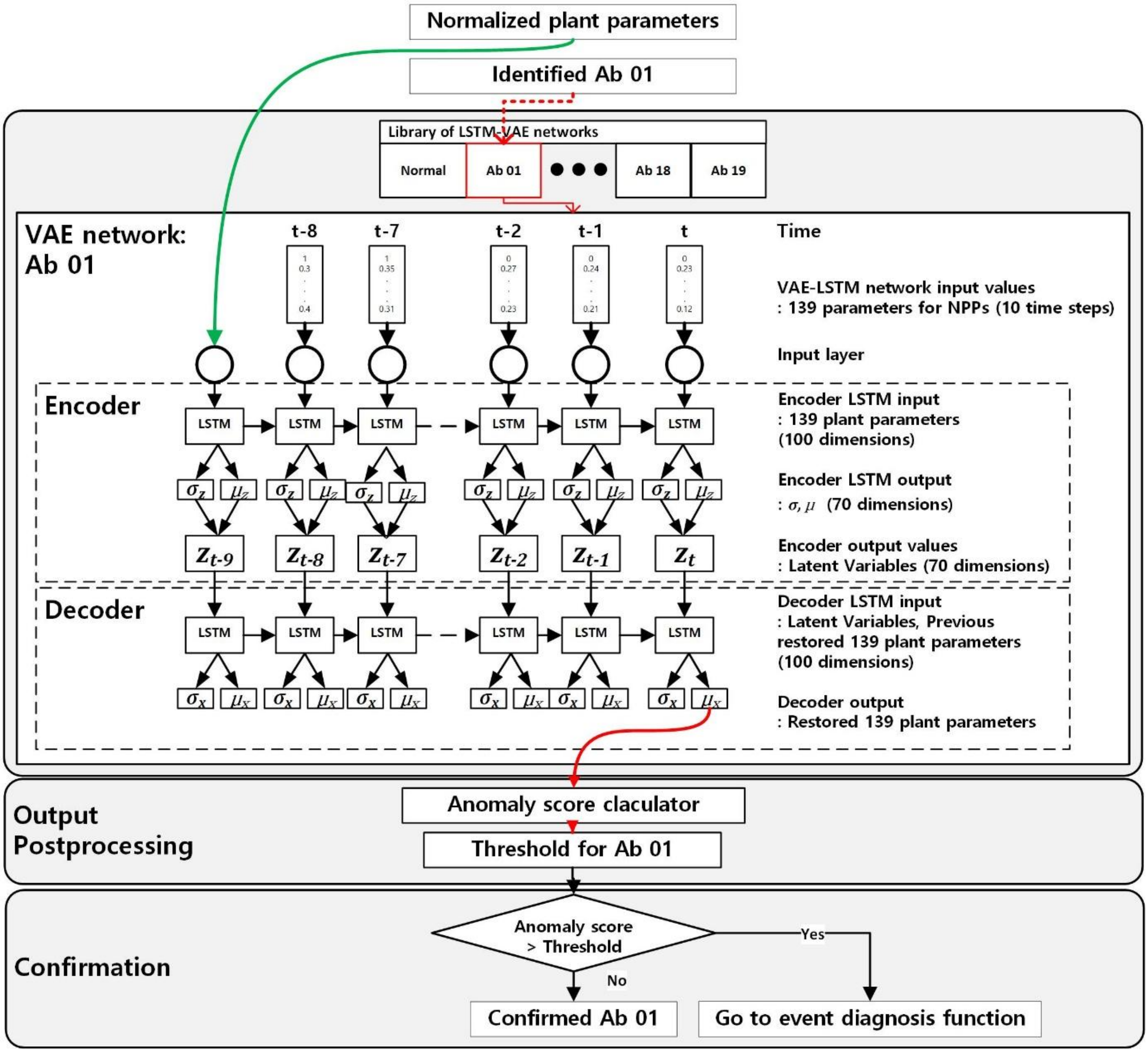

Normalized plant parameters
Identified Ab 01
Library of LSTM-VAE networks
Normal
Ab 01
Ab 18
Ab 19
VAE network: Ab 01
t-8
t-7
t-2
t-1
t
Time
VAE-LSTM network input values
: 139 parameters for NPPs (10 time steps)
Input layer
Encoder
LSTM
σ_z
μ_z
z_{t-9}
z_{t-8}
z_{t-7}
z_{t-2}
z_{t-1}
z_t
Encoder LSTM input
: 139 plant parameters (100 dimensions)
Encoder LSTM output
: σ, μ (70 dimensions)
Encoder output values
: Latent Variables (70 dimensions)
Decoder
σ_x
μ_x
Decoder LSTM input
: Latent Variables, Previous restored 139 plant parameters (100 dimensions)
Decoder output
: Restored 139 plant parameters
Output Postprocessing
Anomaly score claculator
Threshold for Ab 01
Confirmation
Anomaly score > Threshold
Yes
No
Confirmed Ab 01
Go to event diagnosis function

Fig. 17. Illustration of the confirmation of diagnosis results function.

3. Verification

Verifications were performed for 149 scenarios (i.e., 57,309 datasets). Among them, 100 cases and 47,987 datasets were used for the trained events, while 49 cases and 9,322 datasets were used for untrained events. As a result of the verification, the accuracy of the network of unknown event identification (i.e., predicting a trained event as a known event and an untrained event as an unknown event) is 96.72%. In addition, the accuracy of the network for confirmation of diagnosis results is 98.44%. The verification demonstrated that the algorithm could successfully diagnose the trained events and identify the untrained events. Fig. 18 shows an example for the diagnosis of an untrained event (i.e., Ab 20), which is a leakage from the steam generator tubes. In this case, the identification of unknown event function identifies the event as an untrained event because the construction error goes beyond the threshold. Thus, the message “Unknown Event” is provided.

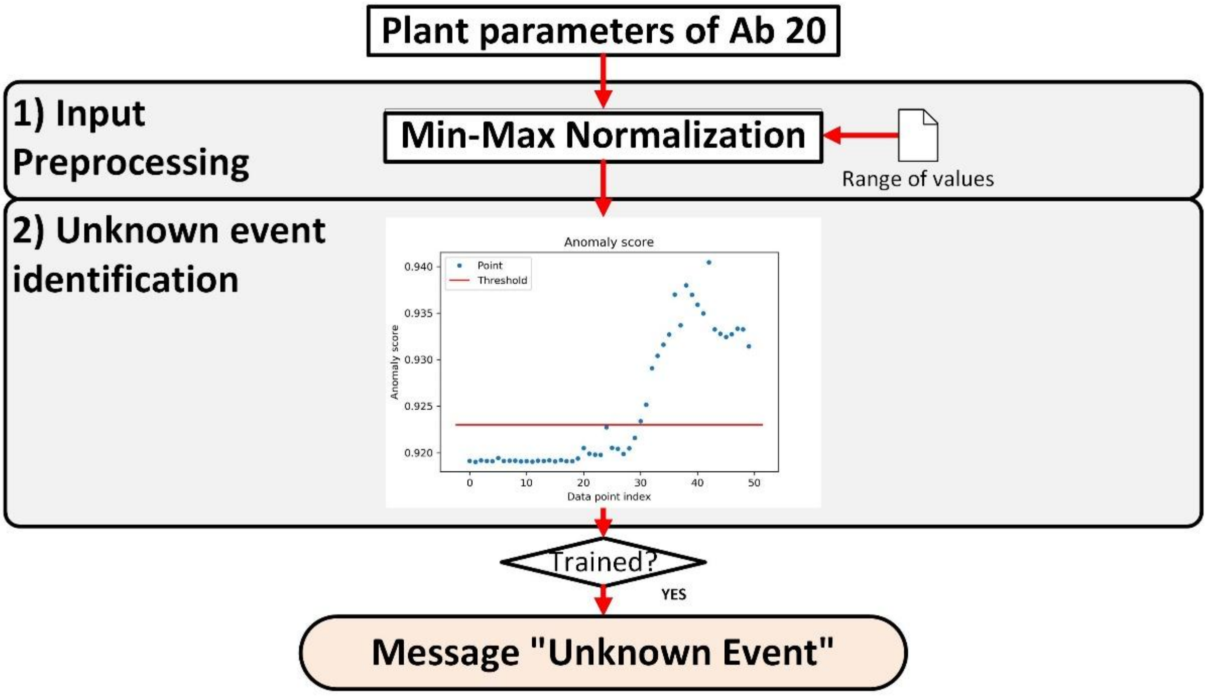


Fig. 18. Process of diagnosing an untrained event.

Fig. 19 illustrates how the algorithm diagnoses a trained event (i.e., event Ab 08), which is the continuous insertion of control rod. The input pre-processing function normalizes plant parameters. Subsequently, the identification of unknown event function identifies the current situation as a trained event. The event diagnosis function examines the event as Ab 08, and diagnosis result function confirms that the diagnosis result is correct. Finally, “Ab 08: continuous insertion of control rod” is presented as the diagnosis result for the current situation.

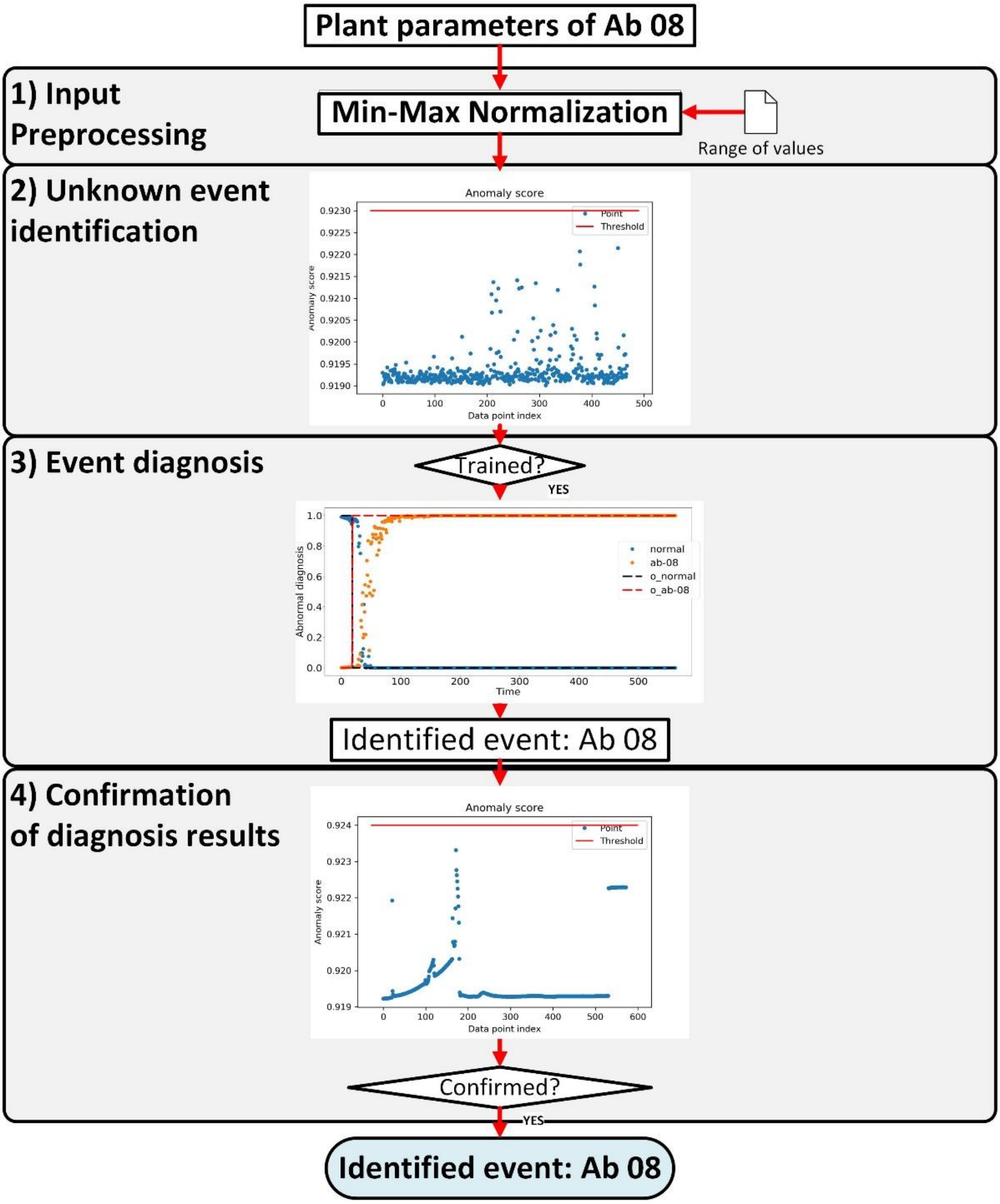


Fig. 19. Process of diagnosing a trained event.

Ⅳ. Development of Long-Term Prediction with Uncertainty Estimation Algorithm

A. Long-Term Prediction Algorithm Design

This section describes the overall structure of the multivariate and long-term trend prediction algorithms for the critical safety parameters. Fig. 20 illustrates the functional architecture of the prediction algorithm design, which comprises four functions: 1) Pre-processing, 2) long-term prediction, 3) uncertainty estimation, and 4) post-processing.

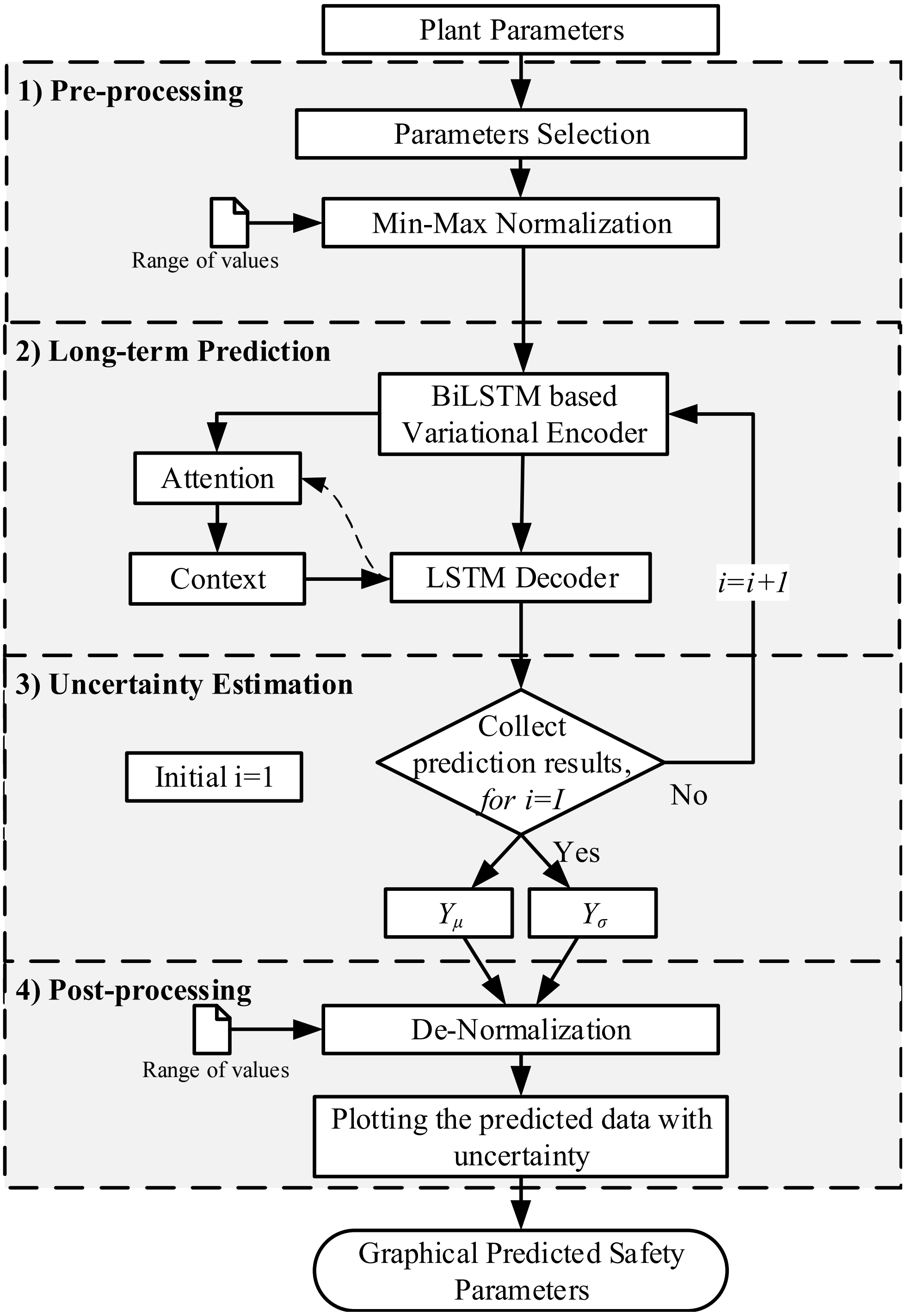


Fig. 20. Functional architecture of long-term prediction with uncertainty estimation algorithm design.

1. Pre-processing Function

The first function of the algorithm is to process the plant parameters and make them suitable as network inputs. The input data for the network should have a range of values from 0 to 1. However, plant parameters have a different range of values or states. Generally, variables with higher values will have a larger impact on the network results. However, higher values are not necessarily more important for prediction. This problem produces local minima. Therefore, the normalization processing obtains the regular plant parameters as input and then outputs the normalized plant parameters that will be utilized by the network.

Min-max normalization is used to prevent the local minima problem and increase the learning speed. The training data of the network is calculated by using Eq. (15). Basically, time-series predictions are strongly dependent on input parameters. Therefore, parameter selection is one of the most important processes in a proposed algorithm. The process to determine the input parameters is discussed in Section Ⅳ.B.3.

2. Long-Term Prediction Function

The long-term prediction function is used to predict the long-term trend of variables for critical safety parameters via BiLSTM and AM networks. This network is based on an encoder - decoder network to apply for the MIMO framework, as well as an applied Bayesian model to estimate the model uncertainty. This network receives the normalized plant parameters from the pre-processing function and predicts multivariable and long-term behaviors. Fig. 21 illustrates the architecture of the long-term prediction function, which comprises four main components: a BiLSTM-based variational encoder, an LSTM decoder, an attention layer, and a decoder combined with the attention layer. For the input multivariate time-series data $X \in R^{T \times D}$ (where T is the number of input time steps, and D is the number of input parameters), the BiLSTM yields the temporal representation of the time-series data via the encoder hidden states $H^{BiLSTM} = \left[h_1^{BiLSTM}, \ldots, h_T^{BiLSTM}\right] \in R^{T \times 2M}$, where each encoder hidden state can be defined as $h_i^{BiLSTM} \in R^{2M}$. The encoder hidden state is combined with the hidden state of the forward LSTM $\overrightarrow{h_i} \in R^M$ and the hidden state of the backward LSTM $\overleftarrow{h_i} \in R^M$ (where M is the number of each LSTM dimension, and i is the encoder i^{th} time step of the encoder). Hence, at the end of the encoder process, we obtained the last hidden state h_T^{BiLSTM}, and the encoder outputs H^{BiLSTM}.

The exact modeling of the true posterior $p(z|x,y)$ is intractable. Subsequently, the last hidden state of the BiLSTM is input to two linear transformation layers to estimate the mean $\mu_z \in R^L$ and standard deviation $\sigma_z \in R^L$ of the latent variable $z \in R^L$ (where L is the number of latent variable dimensions) to determine $q_\phi(z|X, Y)$, which is a parameter for a normal distribution corresponding to the last hidden state h_T^{BiLSTM}.

The latent variable z is used as the input value for the LSTM decoder to yield the decoder hidden state $S=[s_1,\cdots s_N]\in R^{N\times 2M}$, where N is the number of predicted time steps. Therefore, at the decoder time step j, the first step of the attention layer computes the alignment scores e_{ji} using H^{BiLSTM} and s_j, as shown in Eq. (19). e_{ji} is a vector of length T, where each element represents $i\in 1,\ldots T$ the alignment score dedicated to the encoder output. Next, the attention weights $\alpha_{ji}\in R^T$ are computed, as shown in Eq. (20). Subsequently, these attention weights are used in an inner product with H^{BiLSTM} to compute the context vector $context_j\in R^{2M}$ with the respective attention weights $\alpha_{1,i},\cdots,\alpha_{N,i}$, as shown in Eq. (21).

$$e_{ji}=H^{BiLSTM}\bullet s_j\in R^t \tag{19}$$

$$\alpha_{ji}=\frac{\exp(e_{ji})}{\sum_{k=1}^{t}\exp(e_{jk})} \tag{20}$$

$$context_j=\alpha_{ji}\bullet H^{BiLSTM} \tag{21}$$

Next, the attention of the attentional hidden states $h_j^{attention}$ is computed, as shown in Eq. (22).

$$h_j^{attention}=[s_j\oplus context_j]\in R^{4M} \tag{22}$$

Finally, the predicted multivariable output values are obtained by bypassing $h_j^{attention}$ through a decoder combined with an attention layer (i.e., $Y=[y_1,\ldots,y_N]\in R^{N\times D'}$, where N is the number of predicted time steps, and D' is the number of predicted parameters.).

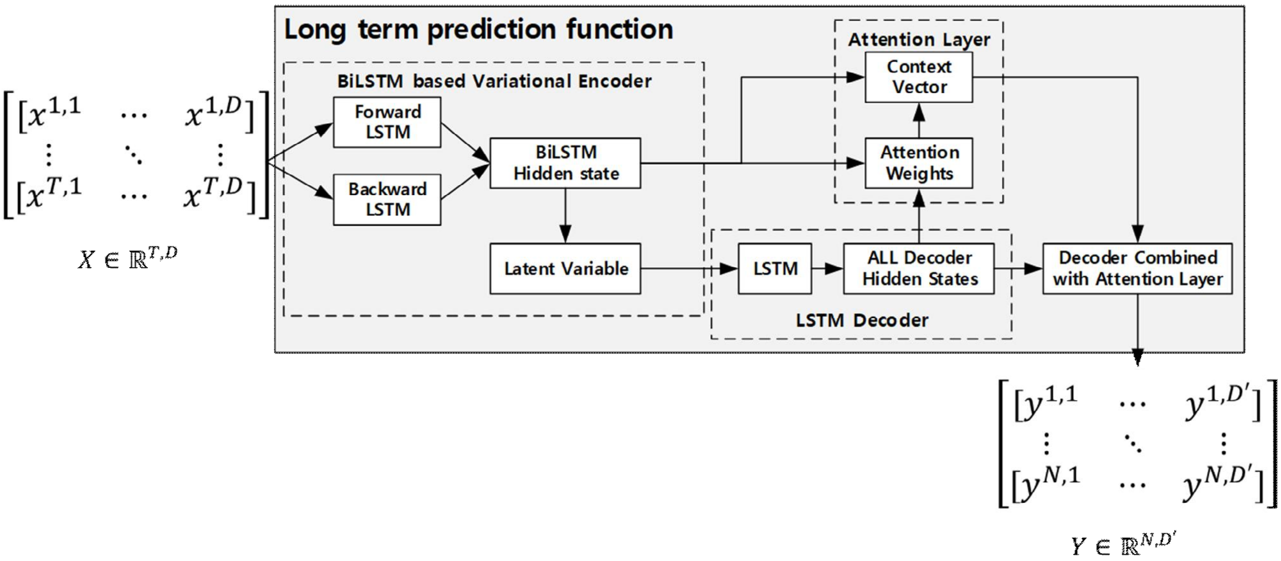

Long term prediction function
BiLSTM based Variational Encoder
Forward LSTM
Backward LSTM
BiLSTM Hidden state
Latent Variable
Attention Layer
Context Vector
Attention Weights
LSTM
ALL Decoder Hidden States
LSTM Decoder
Decoder Combined with Attention Layer
$\begin{bmatrix} [x^{1,1} & \cdots & x^{1,D}] \\ \vdots & \ddots & \vdots \\ [x^{T,1} & \cdots & x^{T,D}] \end{bmatrix}$
$X \in \mathbb{R}^{T,D}$
$\begin{bmatrix} [y^{1,1} & \cdots & y^{1,D'}] \\ \vdots & \ddots & \vdots \\ [y^{N,1} & \cdots & y^{N,D'}] \end{bmatrix}$
$Y \in \mathbb{R}^{N,D'}$

Fig. 21. Process of long term prediction function

3. Uncertainty Estimation Function

The uncertainty estimation function is used to estimate the uncertainty of the long-term prediction results. The proposed network implements the Bayesian model using the VED. It is noteworthy that standard ANNs cannot provide the uncertainty in the prediction results. One of the characteristics of the Bayesian model is that even if the same input data are input to the network, the network outputs different results. Hence, the prediction uncertainty can be quantified by performing several forward passes using the Bayesian model. Fig. 22 illustrates the process of the uncertainty estimation function. This function accumulates each prediction result for $i = 1$ to I, and the mean and standard deviation (SD) are obtained using Eqs. (23) and (24), respectively. To set the confidence interval, the upper bound is determined by adding 1.645 SD to the mean, and the lower bound is determined by adding -1.645 SD to the mean.

$$Y_{\mu} = \frac{1}{I} \Sigma_{i=1}^{I} Y_i \tag{23}$$

$$Y_{\sigma} = \sqrt{\frac{1}{I} \Sigma_{i=1}^{I} (Y_i - Y_{\mu})^2} \tag{24}$$

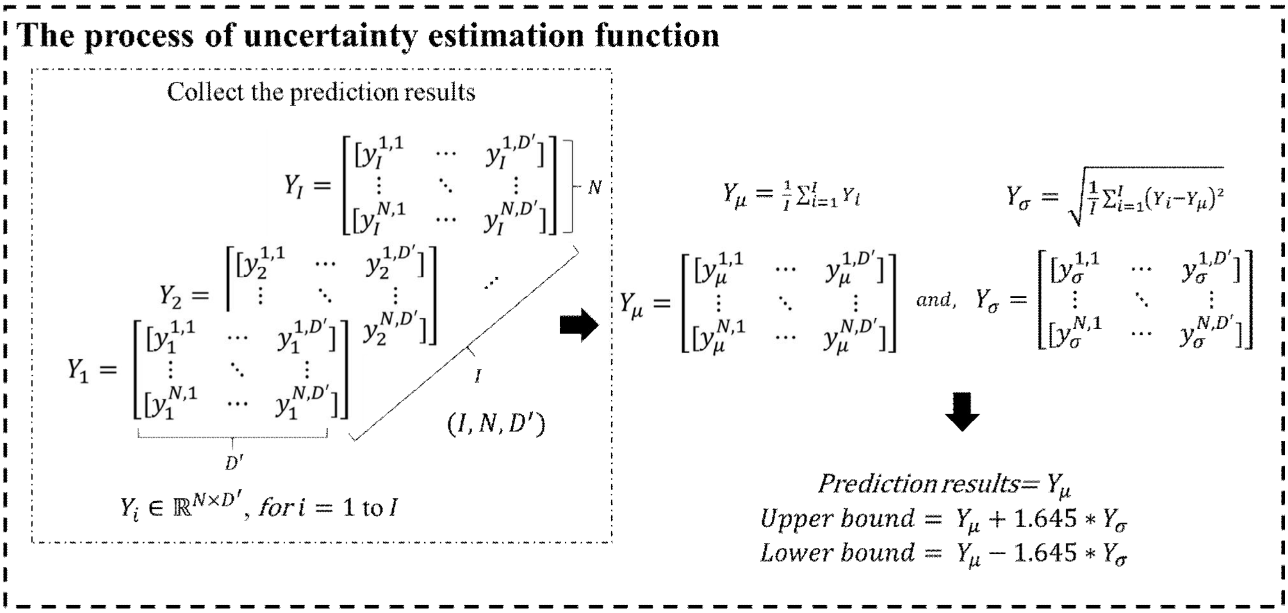

The process of uncertainty estimation function
Collect the prediction results
$Y_I = \begin{bmatrix} [y_I^{1,1} & \cdots & y_I^{1,D'}] \\ \vdots & \ddots & \vdots \\ [y_I^{N,1} & \cdots & y_I^{N,D'}] \end{bmatrix}$
N
$Y_2 = \begin{bmatrix} [y_2^{1,1} & \cdots & y_2^{1,D'}] \\ \vdots & \ddots & \vdots \\ & & y_2^{N,D'}] \end{bmatrix}$
$Y_1 = \begin{bmatrix} [y_1^{1,1} & \cdots & y_1^{1,D'}] \\ \vdots & \ddots & \vdots \\ [y_1^{N,1} & \cdots & y_1^{N,D'}] \end{bmatrix}$
D'
I
(I, N, D')
$Y_i \in \mathbb{R}^{N \times D'}$, for $i = 1$ to I
$Y_\mu = \frac{1}{I}\sum_{i=1}^{I} Y_i$
$Y_\sigma = \sqrt{\frac{1}{I}\sum_{i=1}^{I}(Y_i - Y_\mu)^2}$
$Y_\mu = \begin{bmatrix} [y_\mu^{1,1} & \cdots & y_\mu^{1,D'}] \\ \vdots & \ddots & \vdots \\ [y_\mu^{N,1} & \cdots & y_\mu^{N,D'}] \end{bmatrix}$ and, $Y_\sigma = \begin{bmatrix} [y_\sigma^{1,1} & \cdots & y_\sigma^{1,D'}] \\ \vdots & \ddots & \vdots \\ [y_\sigma^{N,1} & \cdots & y_\sigma^{N,D'}] \end{bmatrix}$
Prediction results$= Y_\mu$
Upper bound $= Y_\mu + 1.645 * Y_\sigma$
Lower bound $= Y_\mu - 1.645 * Y_\sigma$

Fig. 22. Process of uncertainty estimation function.

4. Post-processing Function

The final function is to convert the normalized prediction results into suitable units and ranges to present the variables; subsequently, the results are plotted graphically. The first step of this function is to denormalize the prediction results, i.e., the upper and lower bounds. These data should be transformed into suitable units and ranges for each parameter. Denormalization is performed based on Eq. (25), where Y_{output} and Y are the denormalized and predicted values, respectively. Meanwhile, $Y_{\min}$ and $Y_{\max}$ are the minimum and maximum values of the output train data, respectively. Finally, this function plots the graph using the denormalized mean value and fills the area between the upper and lower bounds for the confidence interval and then presents it to the operators. Fig. 23 shows an example of the plotted graph. The left side of the dotted line represents the past value of the predicted parameter by a blue line. The right side of the dotted line represents the prediction result and confidence interval by the red line and gray area, respectively.

$$Y_{output} = Y_{\min} + Y^{*}(Y_{\max} - Y_{\min}) \tag{25}$$

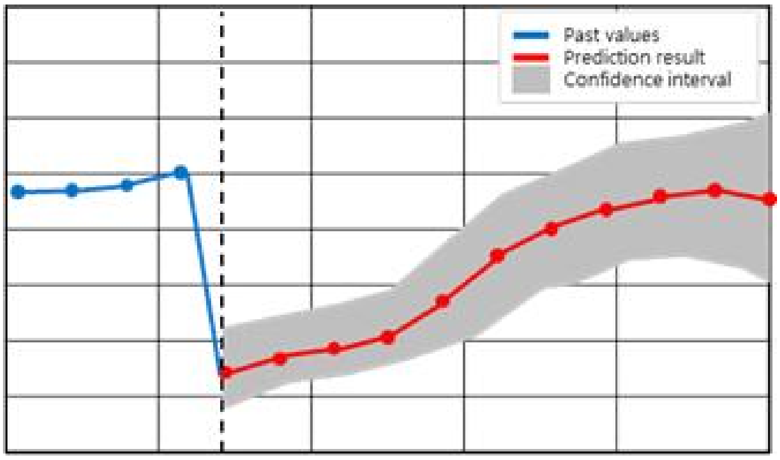


Fig. 23. An example of the plotted graph.

B. Experiment

This study implemented the suggested algorithm by using an NPP simulator. The experiment is performed with data collection, dividing train data, implementation, training and optimization, and verification. For implementation, a desktop computer with NVIDIA GeForce RTX 3090 24GB GPU, Intel 4.00 GHz CPU, Samsung 850 PRO 512GB MZ-7KE512B SSD, and 24GB RAM were used. Python 3.8 was used as a coding language, and several phyton libraries including Keras and Pandas were used to model the algorithm.

1. Data Collection

A sufficient amount of realistic data under emergency situations at the NPP is necessary to ensure the effectiveness of the algorithm. However, emergency situations in real NPPs are scarce. Therefore, the proposed algorithm was implemented using a CNS.

In this study, a loss-of-coolant accident (LOCA) with 16 different break sizes (10 to 25 cm^2 at 1 cm^2 interval) and two leg positions (cold leg and hot leg) were simulated. The total number of scenarios was 32, as shown in Table 4. In all simulations, data were obtained based on the initial plant condition (i.e., 100% full-power normal operation) to shut down the cooling entry condition. Among the 32 cases of the LOCA scenario, 26 cases were used for training, and six cases were used for validating the algorithm.

Table 4. Total collected emergency situation data

Initiating events	Number
Cold leg LOCA	16 (10 to 25cm^2)
Hot leg LOCA	16 (10 to 25cm^2)
Total	32

2. Segregation of Training Data

The segregation of training data involves transforming the data acquired into inputs and outputs to be passed to the proposed network. The input is represented as $X = [x_1, \ldots, x_T] \in R^{T \times D}$, where T is the number of input time steps, and D is the number of input parameters. The output is the predicted multi-step parameters, and it is represented as $Y = [y_1, \ldots, y_N] \in R^{N \times D'}$, where N is the number of time steps of the output, and D' is the number of output parameters. Fig. 24 illustrates the segregation process of the training data. T denotes the length of the input, and N denotes the length of the output. For each time step l of the training datasets, successive data from $l+1$ to $l+T$ form an input sample, and data from $l+T+1$ to $l+T+N$ constitute the corresponding output sample.

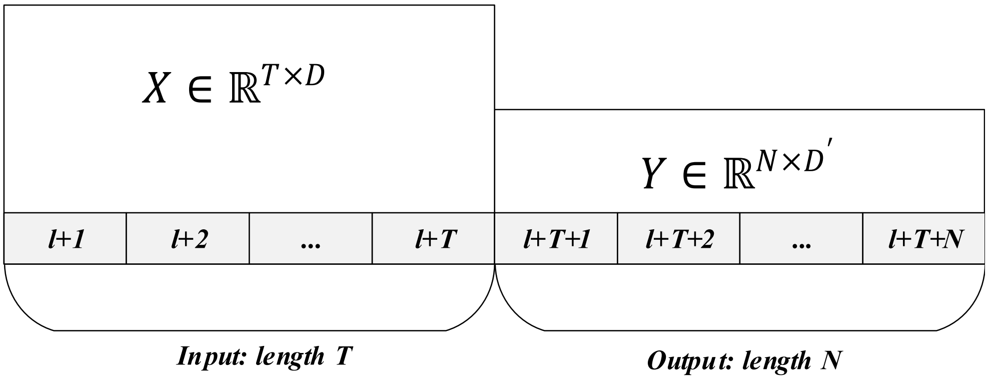

$X \in \mathbb{R}^{T \times D}$
$Y \in \mathbb{R}^{N \times D'}$
l+1
l+2
...
l+T
l+T+1
l+T+2
...
l+T+N
Input: length T
Output: length N

Fig. 24. Segregation process of input data and output data

3. Training and Optimization

Prior to constructing the network, the input and output parameters should be determined. As the output, each selected parameter should be monitored when emergency situations occur at NPPs. Particularly, nine safety functions for satisfying the ultimate goal of NPP safety were identified in Korean NPPs. The nine safety functions were defined to ensure high-level safety objectives and the integrity of NPPs, as well as to prevent the release of radioactive materials; the nine safety functions and their purposes are listed in Table 5 [41, 42]. In this study, we determined 22 total parameters for monitoring based on the nine safety functions. Table 6 lists the output parameters from the CNS.

Table 5. Nine safety functions [41].

No.	Safety function	Purpose
1	Reactivity control	Shut reactor down to reduce heat production
2	Reactor coolant system (RCS) inventory control	Maintain volume or mass of reactor coolant system
3	RCS pressure control	Maintain pressure of reactor coolant system
4	RCS heat removal	Transfer heat out of coolant system medium
5	Core heat removal	Transfer heat from core to a coolant
6	Containment isolation	Close valves penetrating containment
7	Containment pressure and temperature control	Keep from damaging containment
8	Hydrogen control	Control hydrogen concentration
9	Maintenance of vital auxiliaries	Maintain operability of systems needed to support safety systems

Table 6. Output parameters from CNS.

No.	Plant parameter (units)
1	Pressurizer level (%)
2	Pressurizer pressure (kg/cm^2)
3	Steam generator (SG) #1 pressure (kg/cm^2)
4	SG #2 pressure (kg/cm^2)
5	SG #3 pressure (kg/cm^2)
6	SG #1 narrow level (%)
7	SG #2 narrow level (%)
8	SG #3 narrow level (%)
9	Feedwater line 1 flow (kg/s)
10	Feedwater line 2 flow (kg/s)
11	Feedwater line 3 flow (kg/s)
12	Containment pressure (PA)
13	Containment radiation (mRem/hr)
14	Hydrogen concentration (%)
15	Containment temperature (℃)
16	Reactor vessel water level (m)
17	Cold-leg #1 temperature (℃)
18	Cold-leg #2 temperature (℃)
19	Cold-leg #3 temperature (℃)
20	Hot-leg #1 temperature (℃)
21	Hot-leg #2 temperature (℃)
22	Hot-leg #3 temperature (℃)

In general, time-series predictions are dependent on the input parameters. Therefore, input parameter selection is one of the most important processes for the proposed algorithm. The characteristics selected from the used data and the associated meteorological variables that contain the most relevant information must be identified to provide accurate predictions [43]. In this regard, we analyzed the correlations between the output parameters and other parameters before elaborating the prediction algorithm. We used the Pearson correlation coefficient to determine the effects of the parameters on each other.

The Pearson correlation coefficient measures the linear dependence between two random variables. Historically, it is the first formal measure of correlation and is the most widely used measure currently. This linear correlation coefficient, which is used to reflect the linear correlation between two normal continuous variables, is expressed as shown in Eq. (26), where $\overline{x}=\frac{1}{n}\sum_{l=1}^{n}x_l$ and $\overline{y}=\frac{1}{n}\sum_{l=1}^{n}y_l$ denote the mean of x and y, respectively. The achieved correlation coefficients, r_{xy}, are used to represent the correlation between variables, and their values range between −1 and 1.

$$r_{xy}=\frac{\sum(x_l-\overline{x})\sum(y_l-\overline{y})}{\sqrt{\sum(x_l-\overline{x})^2}\sqrt{\sum(y_l-\overline{y})^2}} \tag{26}$$

- $r_{xy}=1$, x and y are a positive correlation,
- $r_{xy}=0$, the linear correlation between x and y is not obvious,
- $r_{xy}=-1$, x and y are a negative correlation.

As 5% Gaussian noise was added to the simulator data to render them similar to actual NPP data, the values of some data might be greater than the maximum value or lower than the minimum value. Therefore, we added a 10% margin to the maximum value and −10% to the minimum value for each parameter. Subsequently, min‐max normalization was performed on the training datasets (i.e., input and output parameters).

As mentioned in Section 4.A, 26 cases (i.e., 5,263 minutes datasets) were trained. A manual search method was used to identify an optimal network by adjusting the absolute values of the Pearson correlation coefficient, $|r_{xy}|$, and the input sequence lengths. Furthermore, no golden rule exists for network optimization [36-38]. Table 7 shows the mean absolute error (MAE), mean square error (MSE), root mean square error (RMSE), mean absolute percentage error (MAPE), and R2 score (R2) for the different networks. The prediction results would be more accurate if the values of the MAE, MSE, RMSE, and MAPE are lower, or if the values of R2 are higher [44-46]. The model evaluation methods utilizing the five indicators are comprehensive and consider the respective emphasis of different indicators. The specific calculations for each indicator are shown in Eqs. 27 - 31, where l and n indicate the index and the number of prediction data, respectively. Y^{real} and Y^{pred} indicate the real and predicted values, respectively, and $\overline{Y^{real}}$ indicates the mean value of Y^{real}.

$$MAE = \frac{1}{n}\sum_{l=1}^{n}\left| Y_l^{real} - Y_l^{pred} \right| \tag{27}$$

$$MSE = \frac{1}{n}\sum_{l=1}^{n}\left(Y_l^{real} - Y_l^{pred} \right)^2 \tag{28}$$

$$RM\mathrm{SE} = \sqrt{\frac{1}{\mathrm{n}}\Sigma_{\mathrm{l}=1}^{\mathrm{n}}\left(\mathrm{Y}_{\mathrm{l}}^{\mathrm{real}} - \mathrm{Y}_{\mathrm{l}}^{\mathrm{pred}}\right)^2} \tag{29}$$

$$MAPE = \frac{100\%}{n}\sum_{l=1}^{n}\left| \frac{Y_l^{real} - Y_l^{pred}}{Y_l^{real}} \right| \tag{30}$$

$$R^2 = 1 - \left[\frac{\sum_{l=1}^{n}\left(Y_l^{real} - Y_l^{pred} \right)^2}{\sum_{l=1}^{n}\left(Y_l^{real} - \overline{Y^{real}} \right)^2} \right] \tag{31}$$

Table 7. Comparison results of various configured networks based on fice indicators.

No.	Input sequence	$\lvert r_{xy} \rvert$	RMSE	MSE	MAE	MAPE	R^2
1	30	0.85	0.0063	8.5578e-05	0.0045	2.6689%	0.9990
2	30	0.90	0.0089	1.2944e-04	0.0061	8.2861%	0.9986
3	30	0.95	0.0065	1.3023e-04	0.0046	3.4975%	0.9984
4	60	0.85	0.0050	6.0543e-05	0.0036	2.4170%	0.9995
5	**60**	**0.90**	**0.0035**	**2.4721e-05**	**0.0026**	**1.5647%**	**0.9997**
6	60	0.95	0.0048	6.5584e-05	0.0035	2.3454%	0.9992
7	90	0.85	0.0056	5.8665e-05	0.0041	2.3961%	0.9992
8	90	0.90	0.0052	4.8128e-05	0.0038	2.2859%	0.9994
9	90	0.95	0.0089	9.3425e-05	0.0052	6.6528%	0.9989

Consequently, since RMSE, MSE, MAE, and MAPE values are lowest as well as R^2 value is biggest, among the various configured networks the input sequence and $\lvert r_{xy} \rvert$ were determined as 60 and 0.90, respectively. The input parameters were determined as 439 parameters using $\lvert r_{xy} \rvert$ larger than 0.90 for each output parameter. The time lengths of the input and output were selected as 60 minutes and 120 minutes, respectively. Thus, the input has 60 sequences and 439 parameters while the output has 120 sequences and 22 parameters. Fig. 25 shows the network for the multivariate and long-term prediction by using the Bi-LSTM based VED with AM. VED provides a variational information bottleneck that prevents overfitting. Therefore, this network does not necessarily tune the hyperparameters [34, 35]. Additionally, this proposed network used Rmsprop [47] optimizer with a learning rate of 0.00001 and ran for 1,000 epochs.

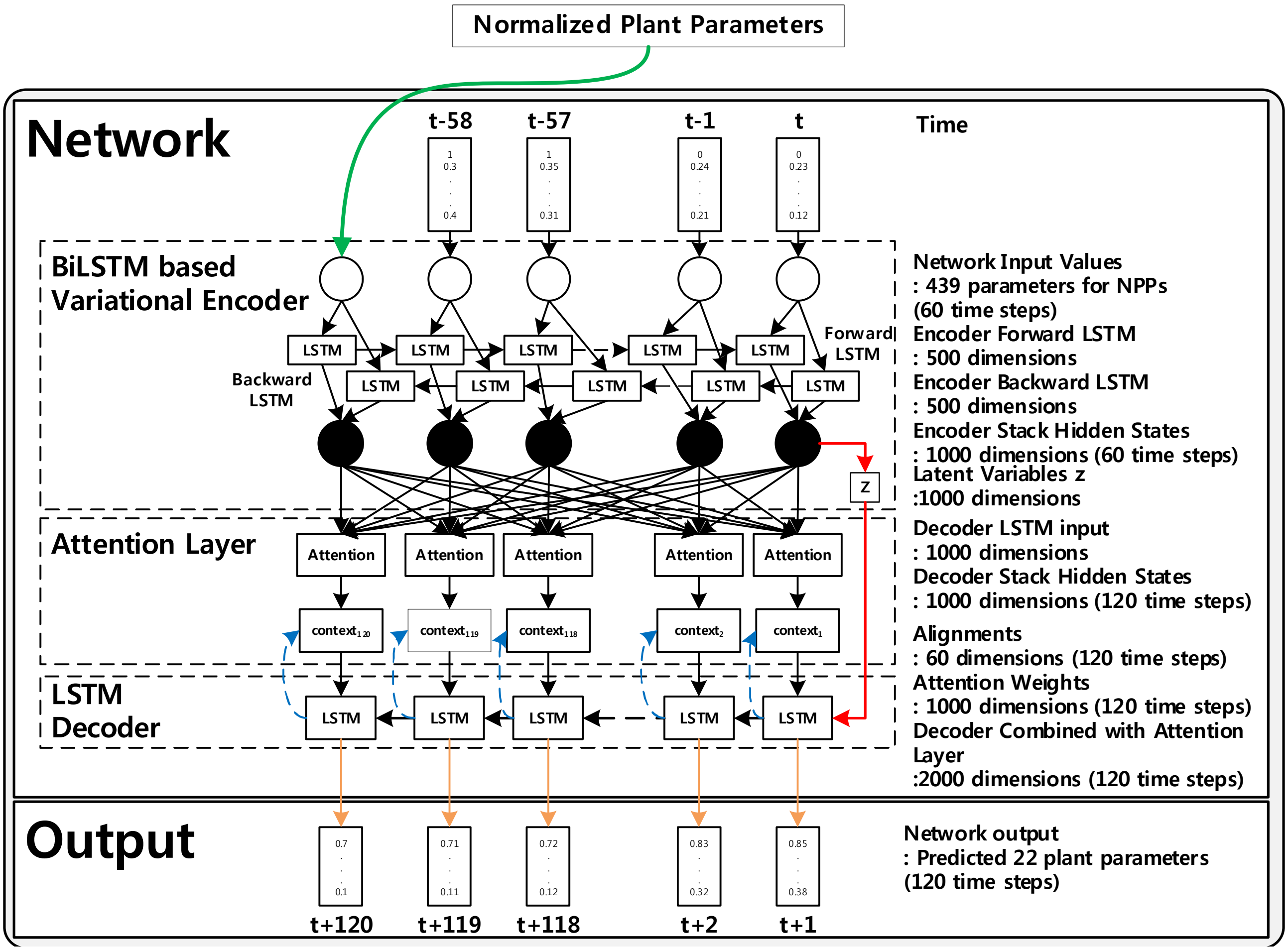


Fig. 25. Illustration of the prediction network.

4. Verification

Verification was performed for 6 test cases (i.e., 1,315 minutes datasets). As a result of the verification, the RMSE, MSE, MAE, MAPE, and R^2 of the network are 0.0137, 1.9001e-04, 0.0076, 3.9377, and 0.9974, respectively. Among the five indicators, a MAPE of more than 50 suggests an inaccurate prediction result, 20-50 is considered a reasonable prediction result, 10 to 20 is considered a good prediction result, and less than 10 is considered a highly accurate prediction result [41]. The verification demonstrated that the proposed network could accurately predict 120 minutes of the safety-critical parameters (i.e., 120 steps and 22 parameters). Furthermore, to estimate the uncertainty, this study collected 100 times of prediction results, and the processing time has been taken 6 seconds.

Fig. 26 shows the test results of 2 hours prediction of safety-critical parameters with uncertainty estimation for the LOCA with sizes of 24 cm^2 in the cold-leg after immediately malfunction injection. The blue line represent the past values of predicted parameters. Orange and red lines represent the real values and the prediction results from the proposed algorithm, respectively. While the gray areas represent the confidence interval. The results show that the proposed algorithm could not only predict accurately parameters immediately after emergency situations occur but also estimate the uncertainty of prediction results.

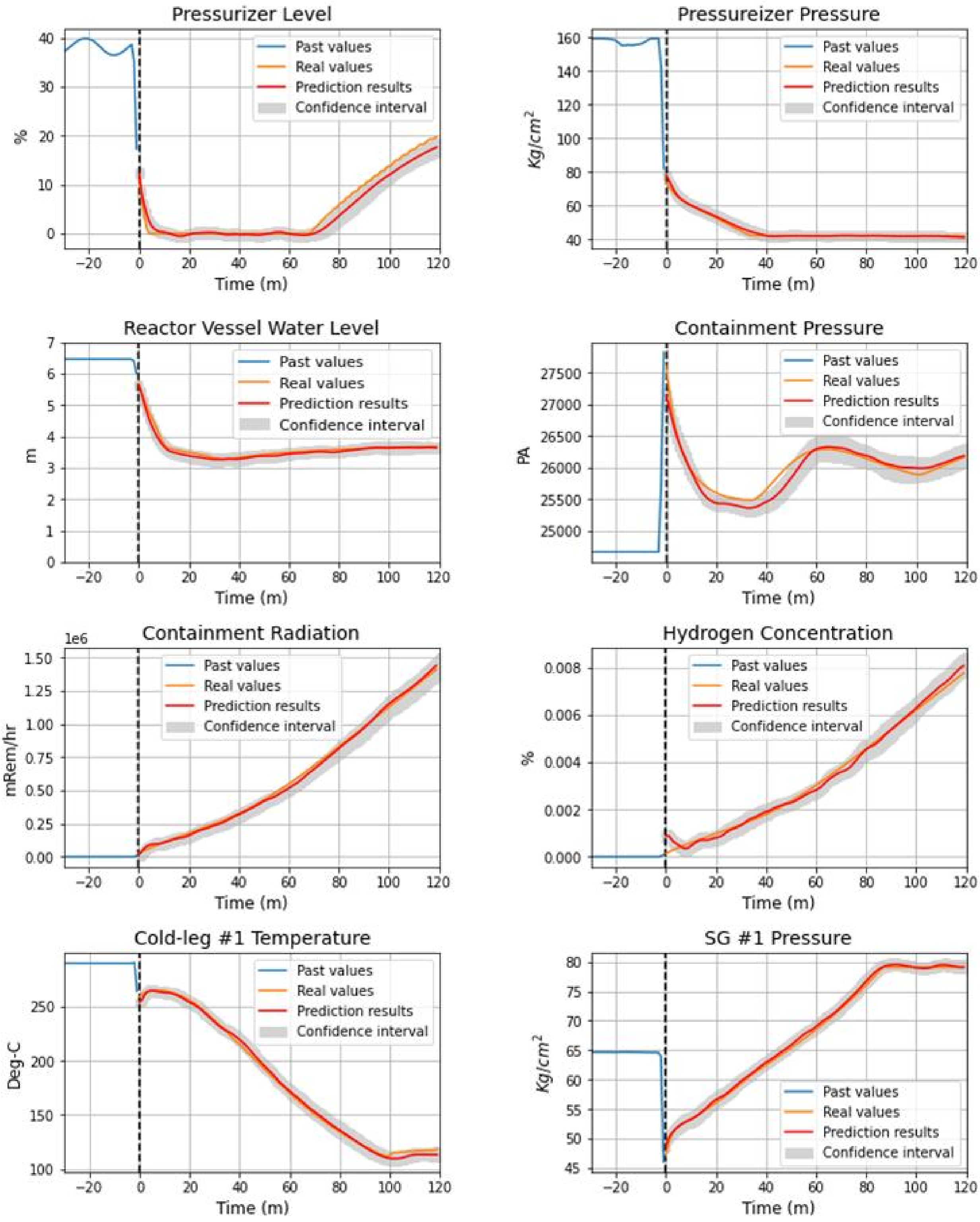


Fig. 26. 2 hours prediction test results of safety-critical parameters with uncertainty estimation under 24cm^2 LOCA in the cold-leg.

Ⅴ. Conclusion

This thesis proposed diagnostic and prognostic algorithms to support operator. Diagnostic algorithm that uses LSTM and VAE networks to diagnose abnormal situations in NPPs was proposed. this algorithm has the capability of finding unknown situations, diagnosing known situations, and confirming the results. The validation demonstrated that the proposed algorithm could provide correct diagnostic results as intended. In addition, the long-term prediction algorithm that uses Bi-LSTM and AM network to predict long-term trends, and provide uncertainty estimation was proposed. The long-term prediction algorithm has not only the capability of predicting 22 parameters and 120 minutes at one time but also provides uncertainty information of prediction results. For a more realistic evaluation, noise-added signals were also considered. The validation demonstrated that the proposed algorithm could provide the accurate prediction as intended. Those algorithms can be applied to an operator support system to improve the operator’s situation awareness during abnormal or emergency situations in NPPs. However, ANN methods have the ‘black box’ problem (ie., understanding how it makes decisions is difficult because of its inability to explain itself). Since the nuclear industry is highly conservative in the adoption of new technologies, future work may consider the use of explainable AI to achieve a similar result as this study. This will enhance the adoption of this algorithm for actual diagnosis of operators in the NPPs.

Reference

[1] IAEA. 2013. “Advanced Surveillance , Diagnostic and Prognostic Techniques in Monitoring Structures , Systems and Components in Nuclear Power Plants.” IAEA Nuclear Energy Series, no. NP-T-3.14.

[2] Kim, Seung Geun, Young Ho Chae, and Poong Hyun Seong. 2020. “Development of a Generative-Adversarial-Network-Based Signal Reconstruction Method for Nuclear Power Plants.” Annals of Nuclear Energy 142: 107410. https://doi.org/10.1016/j.anucene.2020.107410.

[3] Wang, Hang, Min Jun Peng, Peng Wu, and Shou Yu Cheng. 2016. “Improved Methods of Online Monitoring and Prediction in Condensate and Feed Water System of Nuclear Power Plant.” Annals of Nuclear Energy 90 (2016): 44 – 53. https://doi.org/10.1016/j.anucene.2015.11.037.

[4] Bae, Hyeon, Seung Pyo Chun, and Sungshin Kim. 2006. “Predictive Fault Detection and Diagnosis of Nuclear Power Plant Using the Two-Step Neural Network Models.” Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3973 LNCS: 420 – 25. https://doi.org/10.1007/11760191_62.

[5] Park, Jooyoung, Daeil Lee, Wondea Jung, and Jonghyun Kim. 2017. “An Experimental Investigation on Relationship between PSFs and Operator Performances in the Digital Main Control Room.” Annals of Nuclear Energy 101: 58 – 68. https://doi.org/10.1016/j.anucene.2016.10.020.

[6] Park, Jooyoung, Awwal Mohammed Arigi, and Jonghyun Kim. 2019. “A Comparison of the Quantification Aspects of Human Reliability Analysis Methods in Nuclear Power Plants.” Annals of Nuclear Energy 133: 297 – 312. https://doi.org/10.1016/j.anucene.2019.05.031.

[7] Endsley, M. R. 1995. “Toward a Theory of Situation Awareness in Dynamic Systems.” Human Factors 37 (1): 32 – 64. https://doi.org/10.1518/001872095779049543.

[8] She, Jingke, Tianzi Shi, Shiyu Xue, Yan Zhu, Shaofei Lu, Peiwei Sun, and Huasong Cao. 2021. “Diagnosis and Prediction for Loss of Coolant Accidents in Nuclear Power Plants Using Deep Learning Methods.” Frontiers in Energy Research 9 (May). https://doi.org/10.3389/fenrg.2021.665262.

[9] Hsieh, Min Han, Sheue Ling Hwang, Kang Hong Liu, Sheau Farn Max Liang, and Chang Fu Chuang. 2012. “A Decision Support System for Identifying Abnormal Operating Procedures in a Nuclear Power Plant.” Nuclear Engineering and Design 249: 413 – 18. https://doi.org/10.1016/j.nucengdes.2012.04.009.

[10] Kim, Yochan, and Jung, Wondea. 2013. “A Diagnosis Support System for Abnormal Situations of Hanbit Units 3 & 4. Transactions of the Korean Nuclear Society Autumn Meeting

[11] Na, Man Gyun, Won Seo Park, and Dong Hyuk Lim. 2008. “Detection and Diagnostics of Loss of Coolant Accidents Using Support Vector Machines.” IEEE Transactions on Nuclear Science 55 (1): 628 – 36. https://doi.org/10.1109/TNS.2007.911136.

[12] Wang, Wenlin, Ming Yang, and Poong Hyun Seong. 2016. “Development of a Rule-Based Diagnostic Platform on an Object-Oriented Expert System Shell.” Annals of Nuclear Energy 88: 252 – 64. https://doi.org/10.1016/j.anucene.2015.11.008.

[13] Ohga, Yukiharu, and Hiroshi Seki. 1993. “Abnormal Event Identification in Nuclear Power Plants Using a Neural Network and Knowledge Processing.” Nuclear Technology 101 (2): 159 – 67. https://doi.org/10.13182/NT93-A34777.

[14] Yang, Jaemin, and Jonghyun Kim. 2020. “Accident Diagnosis Algorithm with Untrained Accident Identification during Power-Increasing Operation.” Reliability Engineering and System Safety 202 (May 2019): 107032. https://doi.org/10.1016/j.ress.2020.107032.

[15] Yang, Jaemin, and Jonghyun Kim. 2018. “An Accident Diagnosis Algorithm Using Long Short-Term Memory.” Nuclear Engineering and Technology 50 (4): 582 – 88. https://doi.org/10.1016/j.net.2018.03.010.

[16] Gomez Fernandez, Mario, Akira Tokuhiro, Kent Welter, and Qiao Wu. 2017. "Nuclear Energy System's Behavior and Decision Making Using Machine Learning." Nuclear Engineering and Design 324 (September): 27 – 34. https://doi.org/10.1016/j.nucengdes.2017.08.020.

[17] Moshkbar-Bakhshayesh, Khalil. 2019. "Comparative Study of Application of Different Supervised Learning Methods in Forecasting Future States of NPPs Operating Parameters." Annals of Nuclear Energy 132: 87 – 99. https://doi.org/10.1016/j.anucene.2019.04.031.

[18] El-Sefy, M., A. Yosri, W. El-Dakhakhni, S. Nagasaki, and L. Wiebe. 2021. "Artificial Neural Network for Predicting Nuclear Power Plant Dynamic Behaviors." Nuclear Engineering and Technology 53 (10): 3275 – 85. https://doi.org/10.1016/j.net.2021.05.003.

[19] Qiu, Jiayu, Bin Wang, and Changjun Zhou. 2020. "Forecasting Stock Prices with Long-Short Term Memory Neural Network Based on Attention Mechanism." PLoS ONE 15 (1): 1 – 15. https://doi.org/10.1371/journal.pone.0227222.

[19] Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." Neural Computation 9 (8): 1735 – 80. https://doi.org/10.1162/neco.1997.9.8.1735.

[20] Monner, Derek, and James A. Reggia. 2012. "A Generalized LSTM-like Training Algorithm for Second-Order Recurrent Neural Networks." Neural Networks 25: 70 – 83. https://doi.org/10.1016/j.neunet.2011.07.003.

[21] Cui, Zhiyong, Ruimin Ke, Ziyuan Pu, and Yinhai Wang. 2018. "Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-Wide Traffic Speed Prediction," 1 – 11. http://arxiv.org/abs/1801.02143.

[22] Chen, Qian, Wenyu Zhang, and Yu Lou. 2020. "Forecasting Stock Prices Using a Hybrid Deep Learning Model Integrating Attention Mechanism, Multi-Layer Perceptron, and Bidirectional Long-Short Term Memory Neural Network." IEEE Access 8: 117365 – 76. https://doi.org/10.1109/ACCESS.2020.3004284.

[23] Schuster, Mike, and Kuldip K. Paliwal. 1997. “Bidirectional Recurrent Neural Networks.” IEEE Transactions on Signal Processing 45 (11): 2673 – 81. https://doi.org/10.1109/78.650093.

[24] Bishop, C. M. 2006. “Pattern Recognition and Machine Learning.” Springer.

[25] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. “Sequence to Sequence Learning with Neural Networks.” Advances in Neural Information Processing Systems 4 (January): 3104 – 12.

[26] Zhou, Hangxia, Yujin Zhang, Lingfan Yang, Qian Liu, Ke Yan, and Yang Du. 2019. “Short-Term Photovoltaic Power Forecasting Based on Long Short Term Memory Neural Network and Attention Mechanism.” IEEE Access 7: 78063 – 74. https://doi.org/10.1109/ACCESS.2019.2923006.

[27] Bahdanau, Dzmitry, Kyung Hyun Cho, and Yoshua Bengio. 2015. “Neural Machine Translation by Jointly Learning to Align and Translate.” 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 1 – 15.

[28] Luong, Minh Thang, Hieu Pham, and Christopher D. Manning. 2015. “Effective Approaches to Attention-Based Neural Machine Translation.” Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing, 1412 – 21. https://doi.org/10.18653/v1/d15-1166.

[29] Kingma, Diederik P., and Max Welling. 2014. “Auto-Encoding Variational Bayes.” 2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings, no. Ml: 1 – 14.

[30] An, J., and S. Cho. 2015. “Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability.” Special Lecture on IE 2 (1).

[31] Park, Daehyung, Yuuna Hoshi, and Charles C. Kemp. 2018. “A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder.” IEEE Robotics and Automation Letters 3 (3): 1544 – 51. https://doi.org/10.1109/LRA.2018.2801475.

[32] Xu, Haowen, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, et al. 2018. “Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications.” The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018 2: 187 - 96. https://doi.org/10.1145/3178876.3185996.

[33] Le, Hung, Truyen Tran, Thin Nguyen, and Svetha Venkatesh. 2018. “Variational Memory Encoder-Decoder.” Advances in Neural Information Processing Systems 2018-Decem: 1508 - 18.

[34] Tishby, Naftali, and Noga Zaslavsky. 2015. “Deep Learning and the Information Bottleneck Principle.” 2015 IEEE Information Theory Workshop, ITW 2015. https://doi.org/10.1109/ITW.2015.7133169.

[35] Alemi, Alexander A., Ian Fischer, Joshua V. Dillon, and Kevin Murphy. 2019. “Deep Variational Information Bottleneck.” 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, 1 - 19.

[36] Ruder, Sebastian. 2016. “An Overview of Gradient Descent Optimization Algorithms,” 1 - 14. http://arxiv.org/abs/1609.04747.

[37] Kingma, Diederik P., and Jimmy Lei Ba. 2015. “Adam: A Method for Stochastic Optimization.” 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 1 - 15.

[38] Park, Seong Ho, Jin Mo Goo, and Chan Hee Jo. 2004. “Receiver Operating Characteristic (ROC) Curve: Practical Review for Radiologists.” Korean Journal of Radiology 5 (1): 11 - 18. https://doi.org/10.3348/kjr.2004.5.1.11.

[39] Bergstra, James, and Yoshua Bengio. 2012. “Random Search for Hyper-Parameter Optimization.” Journal of Machine Learning Research 13: 281 - 305.

[40] Snoek, J., H. Larochelle, and R. P. Adams. 2012. “Practical Bayesian Optimization of Machine Learning Algorithms.” In Advances in Neural Information Processing Systems, 2951 - 59.

[41] Lee, Daeil, Poong Hyun Seong, and Jonghyun Kim. 2018. “Autonomous Operation Algorithm for Safety Systems of Nuclear Power Plants by Using Long-Short Term Memory and Function-Based Hierarchical Framework.” Annals of Nuclear Energy 119: 287 - 99. https://doi.org/10.1016/j.anucene.2018.05.020.

[42] KHNP. 2014. “APR1400 design description.” Korea Hydro & Nuclear Power Co., LTD 2014.

[43] Jebli, Imane, Fatima Zahra Belouadha, Mohammed Issam Kabbaj, and Amine Tilioua. 2021. “Prediction of Solar Energy Guided by Pearson Correlation Using Machine Learning.” Energy 224: 120109. https://doi.org/10.1016/j.energy.2021.120109.

[44] Donihue, Michael R. 1993. “Evaluating the Role Judgment Plays in Forecast Accuracy.” Journal of Forecasting 12 (2): 81 - 92. https://doi.org/10.1002/for.3980120203.

[45] Armstrong, J. Scott. 2001. “Evaluating Forecasting Methods,” 443 - 72. https://doi.org/10.1007/978-0-306-47630-3_20.

[46] Moreno, Juan José Montaño, Alfonso Palmer Pol, Albert Sesé Abad, and Berta Cajal Blasco. 2013. "Using the R-MAPE index as a resistant measure of forecast accuracy." Psicothema 25 (4): 500-506.

[47] Tieleman, Tijmen, and Geoffrey Hinton. 2012. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural networks for machine learning 4 (2): 26-31.