IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 26, 2020, accepted September 16, 2020, date of publication September 23, 2020, date of current version October 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3026104

Application of a Deep Learning Technique to the Development of a Fast Accident Scenario Identifier

HYEONMIN KIM, JAEHYUN CHO, AND JINKYUN PARK
Korea Atomic Energy Research Institute, Daejeon 34057, South Korea

Corresponding author: Jinkyun Park (kshpjk@kaeri.re.kr)

This work was supported by the Nuclear Research and Development Program Grant from the National Research Foundation of Korea (NRF), funded by the Korean Government, Ministry of Science and ICT, under Grant NRF-2019M2C9A1055906.

ABSTRACT To obtain more accurate results of probabilistic safety assessment (PSA), it is necessary to reflect more complete dynamics of nuclear power plants. In analyzing these more realistic PSA models, numerous thermal-hydraulic code runs should be performed that typically take from a few minutes to several hours. This paper proposes a fast running model using deep learning techniques to obtain plausible accident scenarios while reducing the resources required to conduct PSA. The developed model is built from a conditional autoencoder, and an analysis of its performance is carried out under both trained and untrained ranges. Taking about one second per scenario, the developed model shows about 0.4% and 1.6% error in the trained and untrained ranges, respectively. As a feasibility study, the aggressive cooldown operation under a small break loss-of-coolant accident in the APR1400 plant was considered. The proposed method can reduce uncertainty in PSA and contribute a key technique to dynamic PSA.

INDEX TERMS Deep learning, dynamic PSA, fast running, model uncertainty, parameter uncertainty, probabilistic safety assessment (PSA).

I. INTRODUCTION

According to the ''Operating Experience with Nuclear Power Stations in Member States'' annual report published by the International Atomic Energy Agency (IAEA), 450 nuclear power plants (NPPs) are in commercial operation as of December 31, 2018 [1]. Since 280 units (corresponding to about 62% of all in-operation NPPs) started their commercial operation more than 30 years ago, the public have concerns about the safety of such aging NPPs. Moreover, from the experience of the Fukushima accident, it is obvious that maintaining the risk level of NPPs as low as possible is the most important goal for ensuring their sustainability. For this reason, many efforts have been spent to systematically evaluate the risk level of NPPs, and diverse techniques have been newly proposed or applied to existing approaches as the results of these efforts. One of the representative traditional techniques is probabilistic safety assessment (PSA) or probabilistic risk assessment (PRA), which has been used for quantifying the risk level of NPPs for many decades all over the world.

Without loss of generality, PSA denotes: ''The method or approach (1) provides a quantitative assessment of the identified risk in terms of scenarios that results in undesired consequences (e.g., core damage or a large early release) and their frequencies, and (2) is comprised of specific technical elements in performing the quantification [2].'' This means that the crux of the PSA technique is to identify, as realistically as possible, the plausible accident scenarios with associated frequencies that can cause undesired consequences. The traditional approach to this is to identify a catalog of accident scenarios by investigating the impact of plausible contributors (e.g., hardware failures and human errors) on the risk level of NPPs.

To this end, PSA models are developed by combining event trees (ETs) and fault trees (FTs). In general, an ET consists of diverse event headings with associated branches that originate from a single initiating event. That is, the initiating event is the starting point that triggers various kinds of postulated scenarios in a given NPP. Each plausible scenario can be expressed

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Zhang.

by the sequence of different event headings that are crucial for properly dealing with the progression of the initiating event. Typical event headings include the expected function of the engineered safety features, such as the delivery of a high-pressure safety injection coolant, as well as the required tasks to be done by human operators, such as the rapid cooling of the reactor coolant system by opening valve #1, for example. Consequently, a catalog of plausible scenarios can be identified by the combination of binary branches with respect to each event heading. Following the construction of an ET with respect to a given initiating event, the next step is to identify the success (or failure) criteria for each event heading. For example, in the case of the event heading pertaining to the delivery of coolant as mentioned above, it is important to determine the amount of high-pressure safety injection (HPSI) flow to be provided to the reactor coolant system. To this end, a precise thermal-hydraulic (TH) code such as the Reactor Excursion and Leak Analysis Program (RELAP) or the Multi-Dimensional Analysis of Reactor Safety (MARS) code must be exhaustively run. Following this, the success (or failure) probability of the corresponding ET can be calculated by FTs that represent the interrelations of the causal factors in delivering the HPSI flow as determined by the TH analysis. Finally, a catalog of accident scenarios leading the status of an NPP to undesired consequences is formed with associated probabilities (e.g., conditional core damage probabilities).

Though there are many advantages of this ET/FT methodology, it is natural to anticipate that the traditional PSA technique based on ETs and FTs intrinsically involves sources of high uncertainties, such as parameter uncertainty and model uncertainty [3]–[6]. For this reason, in order to reduce the uncertainty of PSA results, many researchers have proposed diverse methodologies, whether using ET/FT or not [7]–[18], and which usually adopt a two-stage approach: (1) generating a list of plausible scenarios from an ET by considering the source of parameter and model uncertainty, and (2) determining the consequence of each plausible scenario by running precise TH codes. Applications have included reliability calculation for passive systems [19], [20] and a consequence analysis of a dynamic ET [21].

The problem here is that the number of plausible scenarios to be analyzed by precise TH codes drastically increases for a complicated system containing many subsystems or components. In other words, since a huge amount of resources (time and manpower) is necessary to run the whole spectrum of plausible scenarios, it is unrealistic to expect that the existing two-stage approaches are valid in practice. In order to resolve this problem, a deep learning technique is introduced in this study. One of the key benefits expected from deep learning techniques is that it is possible to create a surrogate system that can emulate the response (or output) of a target system within a very short time. From the point of view of a TH code, a deep learning-based surrogate model can quickly emulate the output of the code when a specific input is given. It is noted that, instead of a surrogate model, the term fast running model will be used in this paper because the main objective of the surrogate model is to shorten the time required to get the TH code results.

The organization of this paper is as follows. In Chapter 2, a background of the uncertainties in PSA and an introduction to deep learning models are given. Chapter 3 covers the development of the fast running model considering data characteristics and detailed model design. After that, methods to secure refined data and the performance of the fast running model are described in Chapter 4. Finally, related discussions conclude the paper in Chapter 5.

II. BACKGROUND

This chapter provides background information about this study. A basic description of PSA uncertainties is given in Section 2.1 in order to justify why a deep learning technique is necessary for reducing PSA uncertainties. In Section 2.2, the basics of deep learning are described in order to facilitate the understanding of the deep learning model proposed in this study.

A. KEY TECHNICAL CHALLENGE TO DEAL WITH PSA UNCERTAINTIES

As briefly explained in Chapter 1, in terms of ensuring the operational safety of NPPs, it is critical to obtain credible PSA results to the greatest extent possible. In this regard, the reduction of uncertainties involved in securing PSA results is one of the most important issues [3]. In other words, since the traditional PSA technique usually considers the propagation of accident scenarios by combining the success (or failure) of event headings that represent the status of significant components and/or human responses, it is unavoidable to include diverse assumptions (and approximations) in constructing the PSA model. This strongly implies that the credibility of PSA results can be largely enhanced by dealing with uncertainty sources. In PSA, two kinds of uncertainties, namely parametric uncertainty and model uncertainty, have been generally emphasized for many decades [3]–[5]. The following excerpts would be helpful for understanding the definitions of these two uncertainty sources [6].

- ''Model uncertainty relates to the uncertainty in the assumptions made in the analysis and the models used. Examples of model uncertainty include the assumptions made as to how a reactor coolant pump in a pressurized water reactor would fail following loss of seal cooling and/or injection [...]''
- ''Parameter uncertainty relates to the uncertainty in the computation of the parameter values for initiating event frequencies, component failure probabilities, and human error probabilities that are used in the quantification process of the PRA model.''

For example, let us consider Figure 1 that depicts a part of an arbitrary ET pertaining to the small break loss-of-coolant accident (SLOCA) in NPPs. As previously stated, one of the important PSA results is the catalog of accident scenarios following the occurrence of an initiating event, which consist

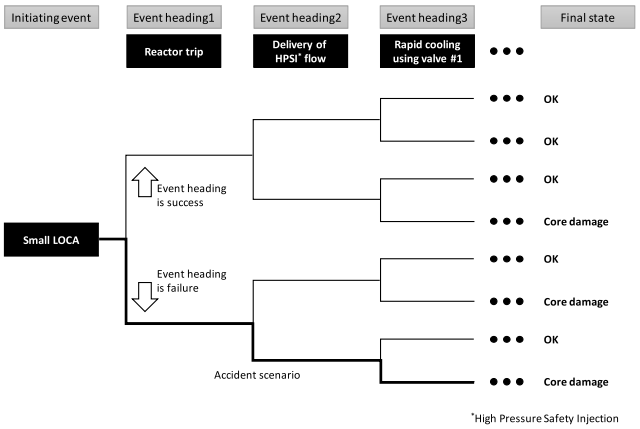

FIGURE 1. Example section of an arbitrary ET.

of the success (or failure) of the event headings. The bold line in Figure 1 highlights one possible accident scenario through the combination of the following event headings: *Reactor trip (Failure) – Delivery of HPSI flow (Failure) – Rapid cooling using valve #1 (Failure)*.

However, the catalog of accident scenarios distinguishable from the same initiating event can be drastically changed by diverse factors such as the status of the related components and the behaviors of the human operators. In this situation, it is more reasonable to assume that the sequence of event headings would become more complicated, as illustrated in Figure 2. In other words, it is highly likely that PSA results contain intrinsic uncertainty originating from the modeling of the event heading sequences.

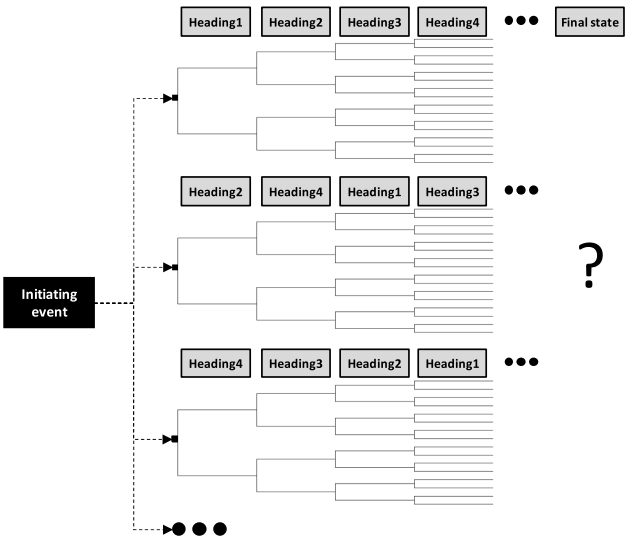

FIGURE 2. Example of an ET to cope with model uncertainties.

Even if we are able to construct several ETs as illustrated in Figure 2, it is still expected that the response of human operators is apt to vary with respect to contextual factors (such as the time available or the amount of information available to perform the required tasks). For example, in terms of the event heading *Rapid cooling using valve #1* in Figure 1, the timing of the valve opening would vary with respect to the nature of the situation the operators are facing. In other words, the variability of human responses could be regarded as a source of parameter uncertainty. This strongly implies that, in reality, the use of multiple branches for each event heading is more reasonable as compared to the use of binary branches shown in Figures 1 and 2. Accordingly, if we want to identify a catalog of accident scenarios that could cover both model uncertainty and parameter uncertainty, it is necessary to consider an ET like the one depicted in Figure 3.

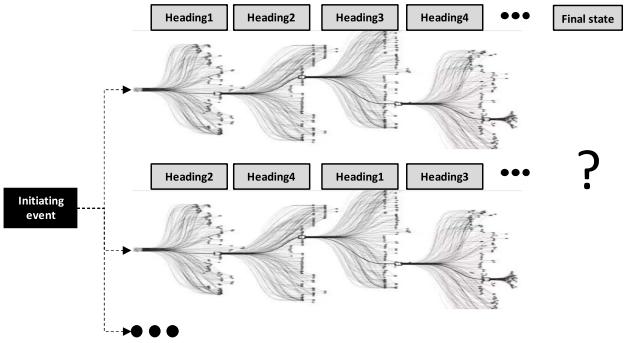

FIGURE 3. Example of an ET to deal with both model and parameter uncertainties [22].

At this level, the challenge becomes identifying the final state of the accident scenarios based on the vast number of TH code runs that must be performed. In other words, since each TH code run takes at least several hours, it is impractical to analyze the whole combination of event headings via precise TH code. Consequently, in terms of practicality, the first technical issue to be urgently resolved is how to determine the final state of the complex combinations of event headings as in Figure 3 within a reasonable time (e.g., a few seconds).

B. INTRODUCTION TO DEEP LEARNING

A deep learning technique able to emulate TH code without prohibitive effort and time is suggested in this work. Before a detailed explanation of the developed deep learning model, this section outlines the concept behind and the training process of a generic artificial neural network (ANN) [23]. The goal of an ANN is to approximate an arbitrary function f^* with a complicated numerical model. While various types of ANNs have been developed, for the sake of basic explanation here, a fully connected network is described. For a classifier, $y = f^*(x)$ maps an input x to the category y. A 3-layer fully connected network defines the mapping $y = f(x; \theta)$ and learns the value of the parameter θ that results in the best function approximation. The general structure of a 3-layer fully connected network is depicted in Figure 4, where $f^{(1)}$,$f^{(2)}$, and $f^{(3)}$ label the respective layers. The combination of several layers gives depth to the model; it is from this terminology that the name ''deep learning'' arises.

The layers between input x and output y are called hidden layers, which consist of nodes (in Figure 4, there are two hidden layers with three nodes in the first hidden layer). Such a model is called a feedforward network because information

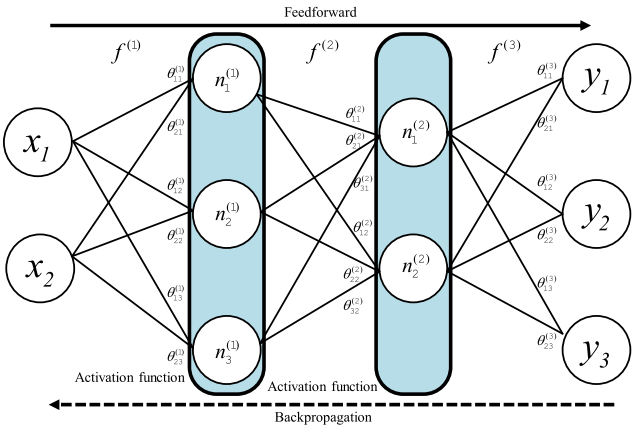

FIGURE 4. Simplified diagram of a fully connected network.

flows through the function being evaluated starting from x, through the intermediate computations used to define f, and finally to the predicted value y. Model training involves calculating the error between the true value (y) and the predicted value ($\hat{y}$), with the errors transferred backward. After that, θ is adjusted and the feedforward process is carried out. Through this iterative process, θ is optimized to reduce error. This process is called backpropagation, and the method to distribute and adjust the error is the gradient descent method [24], as follows:

$$\theta_{t+1} = \theta_t - \eta \frac{\partial E}{\partial \theta} \tag{1}$$

where E is the error between the true value (y) and the predicted value ($\hat{y}$), and η is the learning rate that is able to set the degree of distributed error. The specific type of cost function employed, which calculates the errors, varies depending on the problem. Typically, the cross-entropy cost function is used in classification problems, while mean squared error (MSE) is used in regression problems.

As can be seen in Figure 4, an ANN is typically represented by composing together many different functions. The three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connect in a chain to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. During neural network training, $f(x)$ should be matched to $f^*(x)$. Such multi-layered ANNs have been able to solve, for example, nonlinear problems such as XOR (exclusive or) functions [24]. However, it is difficult to represent more complex nonlinearity using methods with increased depth. To learn complex data (i.e., to represent complex nonlinearity), a nonlinear function is fed into each node called an activation function. The introduced activation function must be computationally efficient to represent complex data, with the choice of activation function depending on the characteristics of the training data; choices include sigmoid, rectified linear unit (ReLU), leaky ReLU, tanh, and others [22], [25]. In addition, neural networks themselves show considerable diversity such as in connection type, which include residual connection (skip connection), and series connection. All such considerations decided by the modeler, such as the activation function, cost function, learning rate, connection type, node configuration are called hyper-parameters, and they are decided by trial and error. The considered hyper-parameters are described below in Chapter 3.

III. DEVELOPMENT OF THE FAST RUNNING MODEL

In Section 2.1, the uncertainty sources of PSA results and associated technical challenge were described. To solve the technical challenge, a deep learning technique called a fast running model is applied, developed in this study based on a conditional autoencoder (cAE). In this chapter, the reason why the authors selected a cAE model for this study is detailed considering the key features of TH datasets. As can be seen from its name, the cAE is a variation of an autoencoder (AE), which is one of the basic structures for creating a generative model. To discover whether a cAE is a reasonable choice for creating the fast running model, it is necessary to clarify: (1) why a generative model is necessary to emulate the results of a precise TH code, and (2) why the cAE is selected among the various generative models.

In terms of developing a deep learning model, similar to all kinds of data-driven models, one critical process is to understand the characteristics of the datasets to be represented by the model. In this regard, the characteristics of the results from a precise TH code (hereafter the term *TH results* will be used), which are needed to analyze the result of each accident scenario, are twofold: time series and autoregressive features. The first characteristic is straightforward because TH results provide the trend of process parameters (e.g., the temperature of a reactor coolant system) as time goes by. Time series is expressed as $X = \{x_1, x_2, \cdots, x_t, \cdots\}$, where x_i are the variables and i is time. The second characteristic, autoregressive data, indicates that the value of the current time step is the output of the previous time step. This feature can be understood by the following equation:

$$x_t = c + \sum_{i=1}^{p} \varphi_i x_{t-i} + \varepsilon_t \tag{2}$$

where $\varphi_1, \cdots, \varphi_p$ are the model parameters, c is a constant, and ε_t is white noise. Generally, in statistical analysis, the autoregressive feature is handled by a moving average model as such as the autoregressive moving average model [26].

If the TH results contain only the time series feature, a general regression model would be enough to emulate them; unfortunately, because of the autoregressive feature of TH results, a general regression model is not a good choice. In general, deep learning models can be thought to belong to either discriminative or generative categories. The major function of the models falling into the first category is to identify a few output classes based on the nature of the input datasets. In contrast, models included in the second category aim to duplicate the key features of the input datasets that are decisive for the regeneration of the input datasets.

Since a generative model defines the distribution of the original data, it is possible to *generate* data similar to the

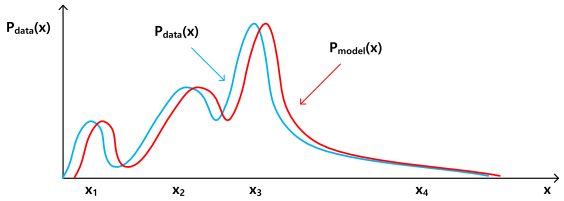

FIGURE 5. Distribution of data in the generative model.

original. For example, let us consider some original data (refer to the blue line in Figure 5) having four features (x_1 to x_4). If x_3 is the most significant feature and x_4 is the least significant feature, it is possible to train a generative model based on x_3 (refer to the red line in Figure 5). In other words, the generative model should be trained so that it follows the characteristics of the significant features as much as possible. In contrast, a discriminative model should be trained so that it can distinguish the existence of the significant features. Accordingly, the accuracy of a discriminative model is generally superior to that of a generative model, if the amount of training data is relatively small or the number of trainings is insufficient. However, if a generative model is well trained with a sufficient amount of data and/or number of trainings that yield a distribution obeying the representative datasets, the generative model is more robust against errors than a discriminative model. In order to handle the autoregressive feature, therefore, the generative model approach is selected here because it will be able to generate an accident scenario by using specific conditions. Indeed, a number of studies have predicted the autoregressive feature by deep learning [27]–[31].

The method of developing a deep learning model can be separated into two trends. The first is producing a dedicated model to learn the features of the input datasets based on mathematical methods such as the convolutional neural network (CNN) or recurrent neural network (RNN), and the second is developing a structural model that learns the data features by modifying the structure of the network. The latter is generally used for generative models. Structural designs that contain wider, deeper, and more complicated connections include the AE and the generative adversarial network (GAN) [32], [33]. Focusing on the characteristics of TH results, it should be emphasized that they depend on diverse conditions expected from each accident scenario, such as the start/stop of specific components, the increase/decrease of important process parameters, and the responses of human operators conducted at different timings. In other words, since the TH result of a given accident scenario is apt to vary with respect to the combination of the abovementioned conditions, it is unrealistic to define a catalog of distinctive classes for the TH result of each accident scenario. Accordingly, the use of a generative model is a reasonable choice in terms of developing a fast running model.

Among a variety of generative models, the two key points why a cAE is designed here are that it can simulate diverse outputs using only a few inputs, and that it can produce credible outputs even in an untrained range. Consider in terms of Figure 3, the inputs indicate the variables that can represent and classify the accident scenario, while the outputs indicate the accident scenario itself. For early stages of the accident, the inputs that can distinguish different accident scenarios are, for example, different temperatures, pressures, and flow rates at the accident location; these represent, however, little information compared to all the information generated in the normal state of an NPP. Moreover, after accidents are classified, the inputs that can distinguish the accident scenario as shown in Figure 3 are only the actions of the event headings. On the other hand, the length of the outputs may increase up to core damage or sufficient marginal time, and in addition, as only those scenarios leading to core damage can be identified, the number of outputs is limited. To check various conditions in dynamic PSA though, a variety of outputs should be considered. In other words, to express a variety of outputs, the number of input is small.

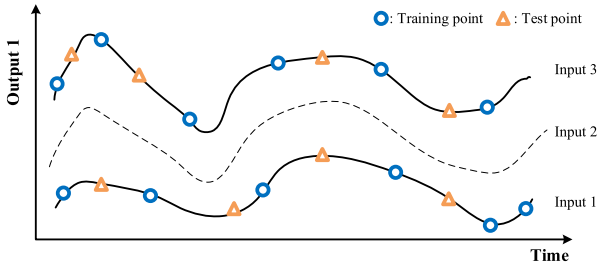

FIGURE 6. Schematic diagram of trained and untrained ranges.

As for the second key point above, the cAE is able to calculate credible results under an untrained range. A conceptual diagram of the trained and untrained ranges is depicted in Figure 6. Model performance here is similar to the general process of test performance in machine learning; i.e., model performance is assessed by splitting datasets into training, validation, and test sets. The training and validation sets are used in model training and hyper-parameter tuning. Generally speaking, all machine learning models should excel in estimating the interpolation problem, represented by specific inputs in Figure 6 (inputs 1 and 3). Here, while the untrained range (input 2) can simply be estimated as the weighted average of inputs 1 and 3, this is not suited to high-dimensionality problems such as TH results because of the high correlation between outputs. To obtain the untrained range, the training and test points for the untrained range should be calculated through TH code. However, in practice, it is not possible to implement the TH code for all inputs. Interestingly, in chemistry, novel chemical structures are generated using a cAE for the reasons mentioned above [34]. Thus, the cAE forms the basis of the fast running model in this work; as the cAE itself is based on an AE and applied conditional techniques, the cAE can be explained by first considering the AE.

The AE is a type of ANN that consists of the same number of input and output nodes, such as the bow-tie model [32].

The key characteristic of the AE is that the data entering the input layer is compressed through the encoder to create a latent space (the latent space is a representative vector from the input data). Thereafter, the decoder is trained to equal the input data via the latent space. Conceptually, the AE has the same function as the principle component analysis (PCA) that can compress data; as the AE extracts feature points through a neural network, though, it is more efficient than PCA when the data is complicated or massive [35]. There are various types of AE, such as the variational autoencoder, denoising autoencoder, and so on [36], [37]. The nuclear field typically calls the AE as an auto-associative neural network (AANN), which has been studied for signal validation and calibration [38], [39].

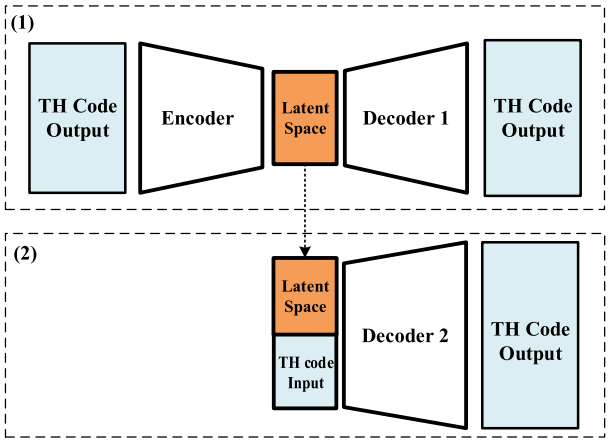

FIGURE 7. Schematic diagram of structure of the developed fast running model.

The fast running model designed in this work is shown in Figure 7. To express output variability using few variables, the model consists of two stages. The first stage generates a latent space representing the feature of the whole dataset; to do so, the input dataset is compressed using a deep learning model (e.g. fully connected model), depicted as 'Encoder' in Figure 7. After that, another deep learning model, designated as 'Decoder' in Figure 7, is trained so that it can regenerate the input dataset based on the given latent space. This process forms the upper panel (1) in Figure 7. However, using the only latent space does not calculate the desired output. For this, the conditional technique can yield the exact results. After calculating the latent space from the first stage, the latent space and the TH code input are combined as input data for the second stage (lower panel (2) in Figure 7). In the second stage, the decoder is trained to calculate the TH results. In other words, the latent space is a vector that can represent the entire dataset; thus, the conditional technique can be used to express variability with only TH code input.

In the application below, the sizes of each sample are 5600, 240, and 2160 (training, validation, and test sets, respectively), giving a total data size of 8,000. In this paper, the simulation length was equally truncated in all cases because AE structure presupposes the same length of inputs and outputs. Though the length of the simulation data differs depending on the break size of the SLOCA and other conditions, the lengths are only truncated such that sufficient time exists for the accident to go to core damage or a safety state. The input data was normalized by a min-max scaler before being split into training, validation, and test sets, as follows. As TH results (e.g., pressure, temperature, and flow rate) have different degrees, changes in the variables are able to effect the model depending on the degree. To avoid this problem, the min-max scaler is applied as below:

$$s = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{3}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of x.

As mentioned above, the deeper and wider a deep learning model is, the better its performance. However, a major point to consider during model design is achieving a simple structure for applicability and extendibility of the developed model. As mentioned above, the cAE has a symmetrical structure. Specifically, in the developed model in this work, the encoder and decoder each have three layers, with each layer having 10, 95, 190, and 380 nodes. The hyper-parameters of the developed deep learning model are summarized in Table 1. It should be noted that the hyper-parameters are not updated through model training; rather, they are defined by the modeler. Thus, to make a fine deep learning model, hyper-parameter tuning should be performed by the validation dataset. In Table 1, as the activation function performing the nonlinearity function in the nodes, ReLU is selected [25].

TABLE 1. Hyper-parameters of the developed deep learning model.

Hyper-parameter	Value
Activation function	ReLU
Optimizer (epsilon)	Adam (0.1)
Learning rate	0.00005
Cost function	MSE
Batch size	10

The MSE is used in this paper, from the following equation:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{4}$$

As ANNs are a type of optimization method, techniques that improve optimization performance should be applied. One of these techniques is the initialization method, as shown in previous research [40]–[42]. In the present work, the He normal initialization method was applied, as expressed in Equations 3.4 and 3.5 [41] below:

$$W \sim N(0, Var(W)) \tag{5}$$

$$Var(W) = \sqrt{\frac{2}{n_{in}}} \tag{6}$$

where n_{in} is the number of nodes in the previous layer. The He normal initialization method is known as having good performance for the ReLU activation function. A moving average with 20 windows is carried out as post-processing.

IV. APPLICATION RESULTS

The developed fast running model was applied to a specific accident scenario of the APR1400 as a feasibility study. Refined datasets and the results of applying the developed deep learning technique to the datasets are described in Sections 4.1 and 4.2, respectively.

A. SECURING REFINED DATA FOR DEEP LEARNING

To generate a sufficient number of input and output datasets of TH code for the application study, a rapid cooldown operation under a small break loss-of-coolant accident (SLOCA) scenario in the APR1400 (Advanced Power Reactor 1400 in Korea) was selected. Data generation was conducted using MARS-KS code, which was developed by the Korea Atomic Energy Research Institute (KAERI) based on the RELAP5/MOD3 and COBRA-TF codes [43]. The APR1400 produces 1400 MWe of electricity at full power operation, and consists of two loops, one steam generator, and two reactor coolant pumps per loop. The SLOCA defines an effective break size of less than about 18.6 cm2, which corresponds to an equivalent diameter of about 1.3 cm to 4.8 cm (0.5–2.0 inch). In SLOCA, cooling water from the in-containment refueling water storage tank (IRWST) or safety injection tank (SIT) is not properly injected early because the pressure of the reactor coolant system (RCS) is higher than the threshold pressure (i.e., the hydraulic head of direct vessel injection pump or SIT injection pressure). Accordingly, it is necessary to reduce the RCS pressure via secondary-side rapid cooldown, which is a strategy to cool the primary side at a speed of 55 K/h by rapidly opening the secondary-side main steam atmospheric dump valve (MSADV).

For the selected accident scenario, four variables and their distributions were defined to conduct a number of TH simulations: MSADV initial open time, RCS cooling rate, reactor coolant pump (RCP) trip time, and duration of available safety injection. It is expected that the earlier the MSADV opening time and the greater the RCS cooling rate, the greater the probability of success of the rapid cooldown operation. The third variable, RCP trip time, is important because not only does the RCP generate significant mechanical heat in comparison to residual heat, but it also affects the RCS coolant circulation regime. Lastly, although the accident scenario defines that safety injection is unavailable, safety injection can either be initially failed (fail-to-start) or initially successful but ultimately fail (fail-to-run).

To obtain probability density functions of the key variables, [44] used main control room simulator data, fault tree models with the failure rates of the components, and expert judgements. Following this approach, the probability density functions of the four variables except the variable 'break size', were determined as follows, and are plotted in Figure 8.

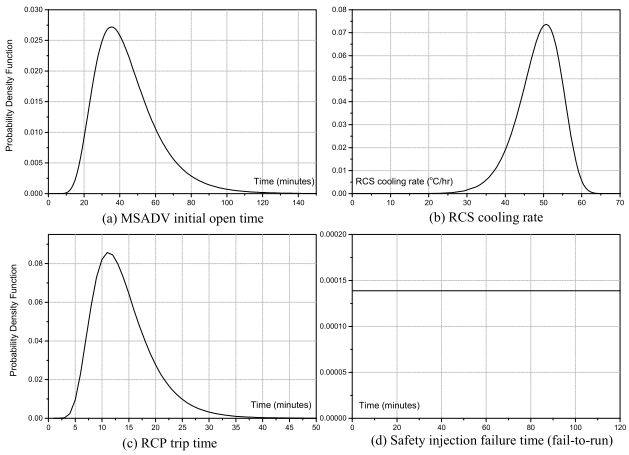

FIGURE 8. Probability density functions of the key variables: (a) MSADV initial open time, (b) RCS cooling rate, (c) RCP trip time, and (d) safety injection failure time.

- MSADV initial open time: log-normal distribution $\ln(X) \sim N(41, 0.385^2)$
- RCS cooling rate: Weibull distribution (alpha: 10.2, lamda: 0.019531)
- RCP trip time: log-normal distribution $\ln(X) \sim N(13, 0.385^2)$
- Safety injection failure time (fail-to-run): $f(t) = 1.61 \cdot 10^{-9} \cdot e^{-1.61 \cdot 10 - 9 \cdot t}$

Monte Carlo sampling and multiple TH simulations were supported by MOSAIQUE code [45]. This code 1) generates sampling inputs for the TH code and 2) runs automatic TH simulations using multiple CPUs. For training data, 2,000 runs each were performed at 0.5, 1.0, 1.5, and 2.0 inch break sizes, producing 8,000 input-output training data. In order to verify the developed deep learning model, 100 runs each were also performed at 1.2, 1.4, 1.6, and 1.8 inch break sizes. Accordingly, the total number of datasets for training and verification was 8,400.

B. REUSULTS

This section gives the results of the performance analysis of the fast running model. The predicted variables were primary side pressure, primary side temperature, secondary side pressure, secondary side temperature, and peak cladding temperature (PCT), which are key variables of accident progression in NPPs. In particular, PCT relates to the degree of fuel damage, with specific PCT values signifying significant damage to the fuel.

The performance analysis is carried out dividing as model test and uncertainty analysis. The model test carries out within trained range, on the other hand, the uncertainty analysis is defined as a performance test about untrained range. If this technology will be used for industrial fields, in all cases, it is hard to obtain the training data for the fast running

model. In the case of these environments, it is essential to assure the performance of untrained data contained within trained data. Thus, performance analysis that consists of the model test and uncertainty analysis is performed. The developed model is able to calculate the results of TH variables for the accident scenarios with less than 3% of the average error in 0.2 s. In comparison, MARS-KS code takes about 2 h to calculate the same results under the same conditions.

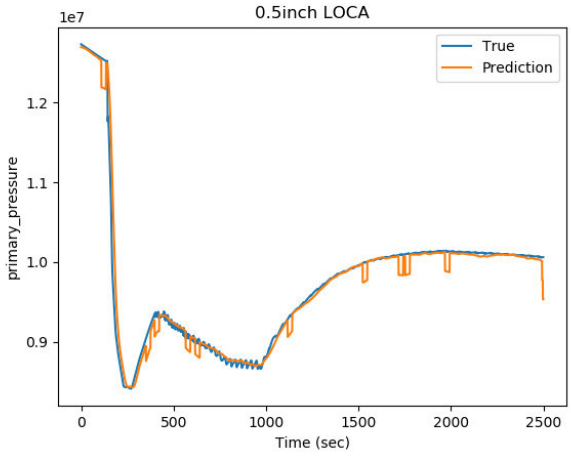

FIGURE 9. Results of the model test for primary pressure, 0.5 inch SLOCA.

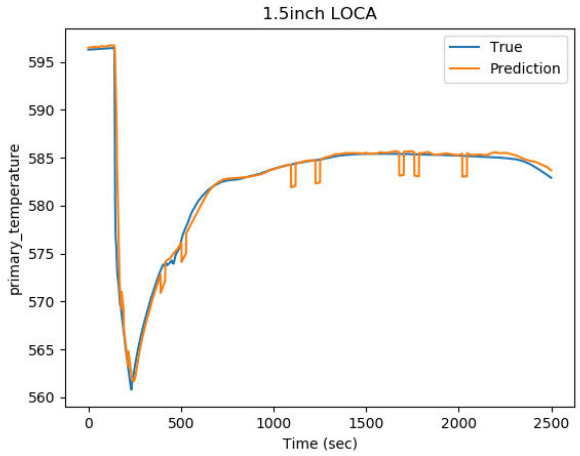

FIGURE 10. Results of the model test for primary temperature, 1.5 inch SLOCA.

TABLE 2. Summary of the model test.

Variable	Mean (%)	Standard Deviation (%)
Primary Pressure	0.59859	0.3062
Primary Temperature	0.08069	0.03659
Secondary Pressure	1.00392	0.54467
Secondary Temperature	0.15451	0.05938
PCT	0.36858	0.51172

The model test produced prediction results for trained data (0.5, 1.0, 1.5, and 2.0 inch SLOCA). Table 2 shows the

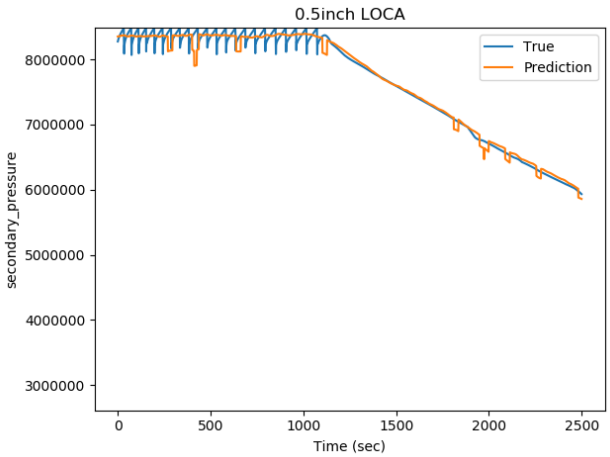

FIGURE 11. Results of the model test for secondary pressure, 0.5 inch SLOCA.

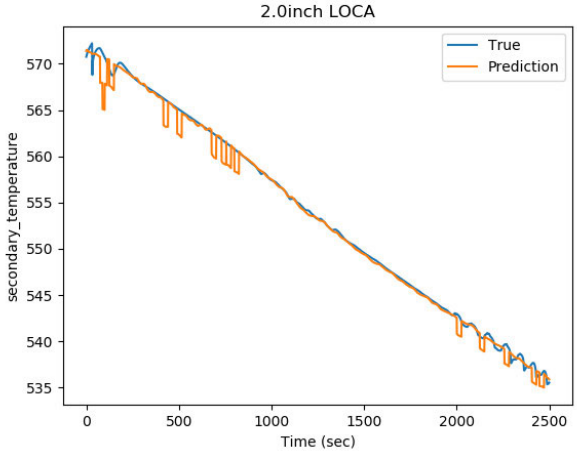

FIGURE 12. Results of the model test for secondary temperature, 2.0 inch SLOCA.

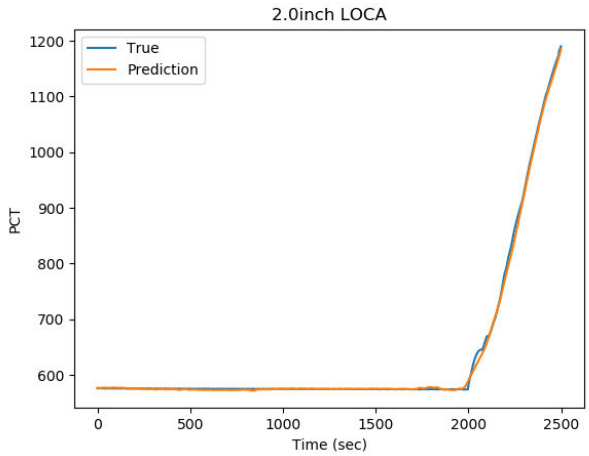

FIGURE 13. Results of the model test for PCT, 2.0 inch SLOCA.

mean and standard deviation values for the accuracy of each variable. Figures 9–13 plot the target values and the predicted values from the fast running model for the four break sizes in

the trained range. In the figures, the blue line is the target value and the orange line is the predicted value.

TABLE 3. Summary of the uncertainty analysis.

Variable	Max Error (%)	Mean (%)	Standard Deviation (%)
Primary Pressure	8.46236	2.48344	1.24801
Primary Temperature	1.60845	0.30775	0.14843
Secondary Pressure	14.55639	3.09434	2.11488
Secondary Temperature	1.5892	0.43361	0.25094
PCT	16.03074	1.83678	2.07727

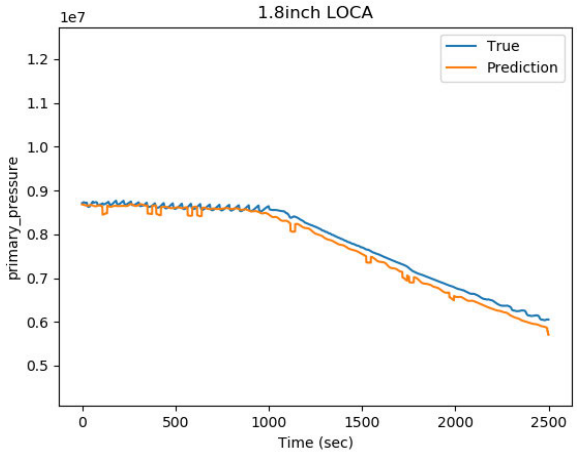

FIGURE 14. Results of the uncertainty analysis for primary pressure, 1.8 inch SLOCA.

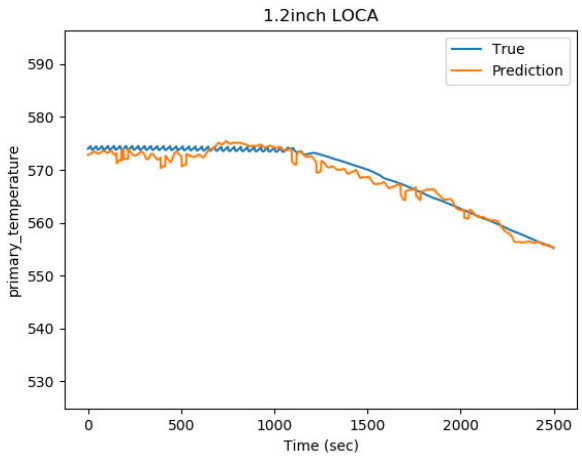

FIGURE 15. Results of the uncertainty analysis for primary temperature, 1.2 inch SLOCA.

The uncertainty analysis considered the prediction of untrained data (1.2, 1.4, 1.6, and 1.8 inch SLOCA). Table 3 summarizes the results of the uncertainty analysis, where max error is included for more information. Though the mean error is 1.6%, there are a few cases having large errors up to about 16%; these should be improved in future studies. Figures 14–18 plot the target and predicted values for the individual cases.

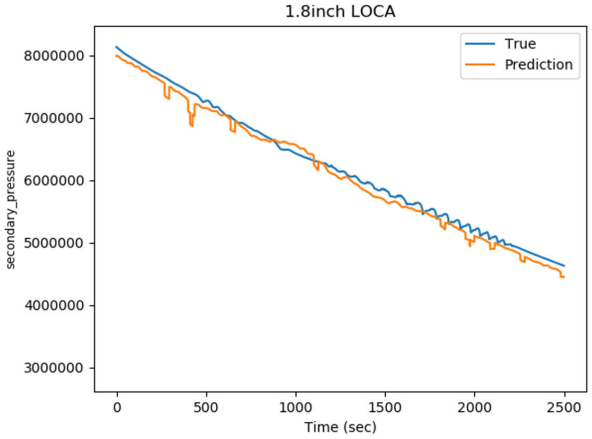

FIGURE 16. Results of the uncertainty analysis for secondary pressure, 1.8 inch SLOCA.

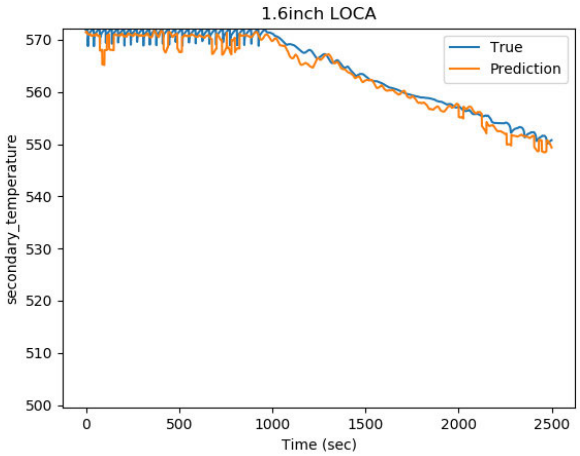

FIGURE 17. Results of the uncertainty analysis for secondary temperature, 1.6 inch SLOCA.

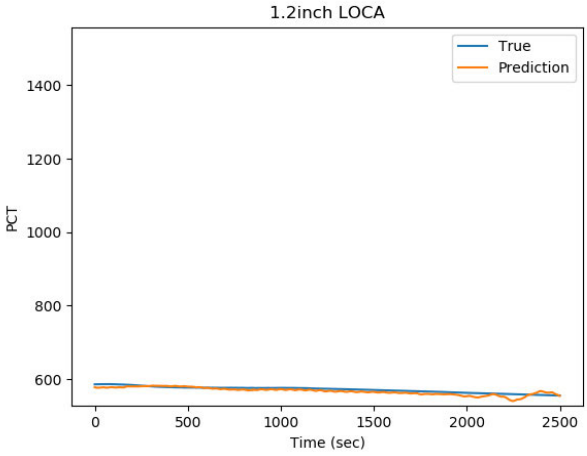

FIGURE 18. Results of the uncertainty analysis for PCT, 1.2 inch SLOCA.

V. DISCUSSION AND CONCLUSION

In order to reduce the enormous code runtimes in obtaining realistic PSA results, a fast running model applying deep learning techniques is suggested in this paper. The traditional

PSA technique based on ETs and FTs has high intrinsic uncertainty, from sources such as parameter uncertainty and model uncertainty. In order to reduce the uncertainty of PSA results, a list of plausible scenarios from an ET is generated and then the consequence of each plausible scenario is determined by running precise TH codes. To analyze more realistic PSA models, numerous code runs should be performed that usually take from a few minutes to several hours. The proposed method addresses this with the ability to calculate TH code with high accuracy in a short time.

A performance analysis of the fast running model was performed under both trained and untrained ranges. Model test results showed an average 0.44% error and a standard deviation of 0.29% error. The uncertainty analysis calculated an average 1.63% error and a standard deviation of 1.68% error, although the largest error was calculated at 16% in one case.

For further studies, model updates will be required. The fast running model developed here is based on an autoencoder, in which the input and the output have the same structural characteristics. The disadvantage then is that data should always be equally entered in all scenarios. Further studies will explore the development of a model that can simulate entire ranges regardless of the length of the scenario. In addition, it is necessary to develop a generative model that can generate a more exact distribution of data than the suggested model. As mentioned in the uncertainty analysis, training data is not able to be generated for all ranges. Therefore, a sampling method will be included that can generate training data through deep learning models.

Applying the fast accident scenario identifier using deep learning can contribute to uncertainty reduction in PSA not only by deriving various plausible accident scenarios but also minimizing the human and computational resources needed to perform the PSA. In addition, it can act as a base technology that can contribute to dynamic PSA, supporting its innovative improvement of NPP safety during operation by deriving optimal response strategies for operators in accident situations.

REFERENCES

[1] International Atomic Energy Agency (IAEA), Vienna, Austria. (Aug. 2019). *Operating Experience With Nuclear Power Stations in Member States*. [Online]. Available: https://www.iaea.org/publications/13575/operating-experience-with-nuclear-power-stations-in-member-states

[2] U.S. Nuclear Regulatory Commission, "An approach for determining the technical adequacy of probabilistic risk assessment results for risk-informed activities," U.S. NRC, Washington, DC, USA, Tech. Rep. 1.200(Part 3), 2004.

[3] T. A. Wheeler, "Treatment of uncertainties associated with PRAs in risk-informed decision making," Sadia Nat. Laboratories, Albuquerque, NM, USA, Tech. Rep. NUREG-1855, 2010.

[4] K. Canavan *et al.*, "Guideline for the treatment of uncertainty in risk-informed applications: Technical basis document," EPRI, Palo Alto, CA, USA, Tech. Rep. EPRI1009652, 2004.

[5] C. K. Park and K.-I. Ahn, "A new approach for measuring uncertainty importance and distributional sensitivity in probabilistic safety assessment," *Rel. Eng. Syst. Saf.*, vol. 46, no. 3, pp. 253–261, Jan. 1994.

[6] K. Canavan *et al.*, "Treatment of parameter and modeling uncertainty for probabilistic risk assessments," EPRI, Palo Alto, CA, USA, Tech. Rep. EPRI1016737, 2008.

[7] M. Čepin and B. Mavko, "A dynamic fault tree," *Rel. Eng. Syst. Saf.*, vol. 75, no. 1, pp. 83–91, Jan. 2002.

[8] J. D. Andrews and J. B. Dugan, "Dependency modelling using fault tree analysis," in *Proc. Syst. Saf. Soc.*, Unionville, MO, Canada, 1999, pp. 67–76.

[9] S. Guarro, M. Yau, and M. Motamed, "Development of tools for safety analysis of control software in advanced reactors," U. S. NRC, Washington, DC, USA, Tech. Rep. NUREG/CR-6465, 1996.

[10] Y. Dutuit, E. Châtelet, J.-P. Signoret, and P. Thomas, "Dependability modelling and evaluation by using stochastic Petri nets: Application to two test cases," *Rel. Eng. Syst. Saf.*, vol. 55, no. 2, pp. 117–124, Feb. 1997.

[11] D. Deoss and N. Siu, "A simulation model for dynamic system availability analysis," M.S. thesis, Dept. Nucl. Eng., MIT, Cambridge, MA, USA, 1989.

[12] C. Acosta and N. Siu, "Dynamic event trees in accident sequence analysis: Application to steam generator tube rupture," *Rel. Eng. Syst. Saf.*, vol. 41, no. 2, pp. 135–154, Jan. 1993.

[13] T. Aldemir, N. Siu, A. Mosleh, P. C. Cacciabue, and B. G. Goktepe, "Automatic generation of dynamic event trees: A tool for integrated safety assessment," in *Reliability and Safety Assessment of Dynamic Process Systems*, vol. 120. Berlin, Germany: Springer, 1994, pp. 135–150.

[14] A. Mosleh and R. Bari, "DDET and Monte Carlo simulation to solve some dynamic reliability problems," in *Proc. PSAM*, New York, NY, USA, 1998, pp. 2055–2060.

[15] E. Hofer, M. Kloos, B. Krzykacz-Hausmann, J. Peschke, and M. Woltereck, "An approximate epistemic uncertainty analysis approach in the presence of epistemic and aleatory uncertainties," *Rel. Eng. Syst. Saf.*, vol. 77, no. 3, pp. 229–238, Sep. 2002.

[16] J. Devooght and C. Smidts, "Probabilistic reactor dynamics—I: The theory of continuous event trees," *Nucl. Sci. Eng.*, vol. 111, no. 3, pp. 229–240, Jul. 1992.

[17] C. Smidts and J. Devooght, "Probabilistic reactor dynamics—II: A Monte Carlo study of a fast reactor transient," *Nucl. Sci. Eng.*, vol. 111, no. 3, pp. 241–256, Jul. 1992.

[18] E. Zio and P. Baraldi, "Identification of nuclear transients via optimized fuzzy clustering," *Ann. Nucl. Energy*, vol. 32, no. 10, pp. 1068–1080, Jul. 2009.

[19] N. Pedroni, E. Zio, and G. E. Apostolakis, "Comparison of bootstrapped artificial neural networks and quadratic response surfaces for the estimation of the functional failure probability of a thermal–hydraulic passive system," *Rel. Eng. Syst. Saf.*, vol. 95, no. 4, pp. 386–395, Apr. 2010.

[20] E. Zio and N. Pedroni, "An optimized line sampling method for the estimation of the failure probability of nuclear passive systems," *Rel. Eng. Syst. Saf.*, vol. 95, no. 12, pp. 1300–1313, Dec. 2010.

[21] *Raven User Manual, Rev. 6*, Idaho Nat. Lab. (INL), Idaho Falls, ID, USA, 2017.

[22] J. Park and H. Kim, "A framework for the dynamic extension and fast progression analysis of accident scenarios using a deep learning technique," in *Proc. Trans. Korean Nucl. Soc.*, Jeju, South Korea, 2019, pp. 1–3.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Boston, MA, USA: MIT Press, 2016.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[25] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, Jun. 2000.

[26] S. E. Said and D. A. Dickey, "Testing for unit roots in autoregressive-moving average models of unknown order," *Biometrika*, vol. 71, no. 3, pp. 599–607, 1984.

[27] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," 2016, *arXiv:1601.06759*. [Online]. Available: http://arxiv.org/abs/1601.06759

[28] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications," 2017, *arXiv:1701.05517*. [Online]. Available: http://arxiv.org/abs/1701.05517

[29] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*. [Online]. Available: http://arxiv.org/abs/1609.03499

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 5998–6008.
[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: http://arxiv.org/abs/1810.04805
[32] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE J.*, vol. 37, no. 2, pp. 233–243, Feb. 1991.
[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2014, pp. 2672–2680.
[34] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules," *ACS Central Sci.*, vol. 4, no. 2, pp. 268–276, Jan. 2018.
[35] P. Baldi and K. Hornik, "Neural networks and principal component anlaysis: Learning from examples without local minima," *Neural Netw.*, vol. 2, no. 1, pp. 53–58, 1989.
[36] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114
[37] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.
[38] J. W. Hines, R. E. Uhrig, and D. J. Wrest, "Use of autoassociative neural networks for signal validation," *J. Intell. Robot. Syst.*, vol. 21, no. 2, pp. 143–154, Feb. 1998.
[39] P. F. Fantoni and A. Mazzola, "Multiple-failure signal validation in nuclear power plants using artificial neural networks," *Nucl. Technol.*, vol. 113, no. 3, pp. 368–374, Mar. 1996.
[40] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. AI Statics*, Sardinia, Italy, 2010, pp. 249–256.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1026–1034.
[42] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 9–48.
[43] J. Jeong, K. S. Ha, B. D. Chung, and W. J. Lee, "Development of a multidimensional thermal-hydraulic code (MARS 1.3.1.)," *Ann. Nucl. Energy*, vol. 26, no. 18, pp. 1611–1642, Dec. 1999.
[44] J. Cho, Y. Kim, J. Kim, J. Park, and D. S. Kim, "Realistic estimation of human error probability through Monte Carlo thermal-hydraulic simulation," *Rel. Eng. Syst. Saf.*, vol. 193, pp. 1066–1073, Jan. 2020.
[45] H.-G. Lim, S.-H. Han, and J. J. Jeong, "MOSAIQUE–a network based software for probabilistic uncertainty analysis of computerized simulation models," *Nucl. Eng. Des.*, vol. 241, no. 5, pp. 1776–1784, May 2011.

HYEONMIN KIM received the B.S., M.S., and Ph.D. degrees in nuclear engineering form Kyung Hee University, Yongin, South Korea, in 2011, 2012, and 2016, respectively. He is currently a Senior Researcher with the Korea Atomic Energy Research Institute (KAERI). His research interests include integrating probabilistic safety assessment and machine learning to improve nuclear safety.

JAEHYUN CHO received the B.S. degree in nuclear engineering and the Ph.D. degree in energy system engineering from Seoul National University, Seoul, South Korea, in 2008 and 2013, respectively. He is currently working with KAERI as a Senior Researcher. His research interests include severe accident probabilistic safety assessment, thermal-hydraulic simulations for success criterial analysis, and passive system reliability.

JINKYUN PARK received the B.S. degree in nuclear engineering from Hanyang University, Seoul, South Korea, in 1991, and the M.S. and Ph.D. degrees in nuclear engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1994 and 1998, respectively.

He joined the Korea Atomic Energy Research Institute (KAERI) as a Researcher, in 2000, where he is currently working as a Principal Researcher. His research interests include probabilistic safety assessment, human reliability analysis, and human performance analysis under diverse off-normal conditions.