Probabilistic deep learning based fast running model of thermal-hydraulic code

Seunghyoung Ryu [a], Hyeonmin Kim [b], Seung Geun Kim [a], Kyungho Jin [b], Jaehyun Cho [b], Jinkyun Park [b*]
[a]*Applied Artificial Intelligence Lab, Korea Atomic Energy Research Institute, 111, Daedeok-daero 989beon-gil, Yuseong-gu, Daejeon 34057, South Korea*
[b]*Risk and Environmental Safety Research Division, Korea Atomic Energy Research Institute, 111, Daedeok-daero 989beon-gil, Yuseong-gu, Daejeon 34057, South Korea*
[*]*Corresponding author: kshpjk@kaeri.re.kr*

1. Introduction

Probabilistic safety assessment (PSA) is a methodology that quantifies the risk of nuclear power plants (NPPs) by evaluating the probability of nuclear accident from a sequence of events. The event refers an initiating event or operator/system's action, and the sequence of events is called a scenario. The probability of accident (e.g., core damage) of corresponding scenario can be derived by simulating thermal-hydraulic dynamics of nuclear reactor. In doing this, thermal-hydraulic (TH) code is used for the simulation, which takes more than hours for the simulation of one scenario.

In order to improve the nuclear safety, the accuracy of PSA can be improved by analyzing broader and granular accident scenarios [1]. Compared to typical binary event tree, degree and timing of action can be considered in dynamic PSA (DPSA). However, long computation time of TH code is critical for massive analysis, thus the development of a fast surrogate model of TH code that approximate the result is necessary.

In this research, we build data-driven surrogate model of TH code via deep learning. First, we formulate the operation of TH code into machine learning problem. Second, we propose LSTM based probabilistic deep learning model as a fast running model of TH code.

Deep learning deals data in batch manner thus the proposed model speeds up the calculation time by generating output of multiple input scenarios at once.

2. Problem Formulation

In this section, we formulate the operation of TH code into machine learning problem. Let $\mathrm{x} = [x_1, x_2, \dots, x_d] \in \mathbb{R}^d$ be the input vector of TH code, where x_i is a value that explains an event. For example, x_1 indicates type of initiating event. x_2 and x_3 represents the degree and time of action for opening charging valve. Then the output of TH code Y is multivariate time-series data in $\mathbb{R}^{T\times N}$ where *T* and *N* are the length and the number of variables to be simulated. Hence, operation of TH code can be formulated as a vector to matrix function, $\mathrm{Y} = f(\mathrm{x})$ which can be modeled by simple multi-layer perceptron.

To leverage temporal relation in output data, this can be also solved with recurrent neural networks (RNN) by reformulating it to matrix to matrix function (so called many-to-many problem). By stacking x for *N* times, we could obtain $\mathrm{X} \in \mathbb{R}^{T\times d}$, where $\mathrm{x}_t = \mathrm{x}\ \forall t$. Based on above setting, we can build RNN as $y_\mathrm{t} = g(\mathrm{x}_t, \mathrm{h}_{t-1})$ where h_{t-1} is hidden states of previous time-step.

3. Deep Learning Model

Based on the problem formulation in Section 2, we build RNN model by utilizing bi-directional LSTM, positional encoding, multiple quantile regression, and model ensemble. Below subsection describes each techniques in brief.

3.1 Bi-directional LSTM.

An RNN is a class of neural network architecture that outputs based on not only the input from the current time-step but also the hidden states from the previous time-step. Long short-term memory (LSTM) [2] is one of classic RNN architecture that resolves long-term dependency problem (i.e., LSTM could learn the relation of data in distant past). LSTM controls the flow of information via gates. According to the direction of information flow in LSTM, it can be classified directional or bi-directional LSTM. Modeling in Section 2 make possible to use of bi-directional LSTM [3] because the action in future is already described in the scenario. LSTM in forward direction learns $y_{f,\mathrm{t}} = f(\mathrm{x}_t, \mathrm{h}_{t-1})$ and in backward direction learns $\mathrm{y}_{b,\mathrm{t}} = f(\mathrm{x}_t, \mathrm{h}_{t+1})$; $\mathrm{y}_\mathrm{t} = g(y_{f,t}, y_{b,t})$.

3.2 Positional encoding

Positional encoding (PE) was introduced for transformer architecture to capture an order-related feature in text data [4]. PE encodes real-value time-step t into a vector $\mathrm{p}_\mathrm{t} \in \mathbb{R}^{d'}$ with sinusoidal function.

$$p_{i,t} = \begin{cases} \sin\left(\frac{t}{10000^{i/d}}\right), & for\ i\ is\ even \\ \cos\left(\frac{t}{10000^{i/d}}\right), & for\ i\ is\ odd \end{cases} \quad (1)$$

In general, p_t has same dimension to x_t and their addition is used as an input. We concatenate two vectors rather than addition. Hence the model could learn temporal feature by explicitly given p_t.

3.3 Multiple quantile regression.

In order to deal with the uncertainty of prediction result, deep learning model could perform probabilistic

forecasting where the model outputs not only the point prediction (mean) but also the distribution information. Two approaches for modeling the output distribution are a) modeling the variance by minimizing Gaussian negative log likelihood and b) modeling multiple quantiles by minimizing pinball loss.

In this research, we formulate the proposed model to predict multiple quantiles simultaneously. Quantile function is the inverse function of cumulative distribution function (CDF). For a random variable X and its cumulative distribution function $F_X(x)$, the τ-th quantile x_τ is $F_X^{-1}(\tau)$ where τ exists between (0,1). For example, if $\tau = 0.2$, the probability of $X \leq x_{0.2}$ is 20 %.

Proposed deep learning model outputs the prediction of $0.1, 0.3, 0.5$ (median), $0.7,$ and 0.9 –th quantiles, then prediction uncertainty can be modeled. The model is trained with pinball loss function. By changing the notation from *x* to *y*, loss for τ-th quantile is

$$L_\tau(y, \hat{y}_\tau) = \begin{cases} \tau(y - \hat{y}_\tau) & if\ y \geq \hat{y}_\tau \\ (\tau - 1)(y - \hat{y}_\tau) & if\ y < \hat{y}_\tau \end{cases} \quad (2),$$

and multiple quantile loss is $\sum_\tau L_\tau$, for $\tau = \{0.1., 0.3, 0.5, 0.7, 0.9\}$.

3.4 Model ensemble and system setup

We trained five model with different random seed for weight initialization. The final output is determined by the average their results. The network is built using *tensorflow*, and trained with adam and learning rate of 0.001. Maximum epoch is set to 150 with reducing learning rate on plateau and early stopping techniques.

4. Experiments

4.1 Model structure

The proposed model namely ensemble quantile recurrent neural network (eQRNN) has following structure.

Input(16) - Tile(16,360) - cPE(24,360) - BiLSTM(256) - BiLSTM(256) - Dense(128) - Dense(64) - Dropout(0.2) - Dense(5)

Tile layer performs tiling operation that expand input vector into matrix. cPE indicates concatenative PE layer with $d' = 8$. BiLSTM, and Dense represent bidirectional LSTM and fully-connected layer, respectively. The numbers in parenthesis are either shape of layer output (Input, Tile, cPE) or key layer parameter (BiLSTM, Dense, Dropout).

4.2 Experimental results

First, we gather the TH code simulation data of total 104,625 scenarios, then divide into training (58%), validation (8%), and two test sets namely test in set (17%) and test out set (17%), respectively. Input scenario is 16 dimensional vector and test out sample corresponds to the scenarios with pressurizer heater on timing of 3.6. Output of TH code is sequential observation of key reactor parameters; 10-seconds interval observations of total 3600-seconds long sequence after initiating event. To evaluate the accuracy of approximation via deep learning model, prediction errors are compared in terms of mean absolute percentage error (MAPE) and mean squared error (MSE). Note that MAPE and MSE is calculated per sequence based on the following equations where y_t and $\hat{y}_t$ represent real and predicted value at time *t*, respectively.

Table I :.Error metrics

MAPE	$\frac{1}{360}\sum_{t=0}^{360}\frac{y_t - \hat{y}_t}{y_t}$
MSE	$\frac{1}{360}\sum_{t=0}^{360}(y_t - \hat{y}_t)^2$

We compare the proposed methodology with two baseline neural networks (FNN : fully-connected neural networks with three dense layers, RNN : two layered bi-directional LSTM with two dense layers). Table II describes the error result of prediction on pressurizer pressure.

Table II: Error results on prediction of pressurizer pressure.

Model	**MAPE**				**MSE**			
	In		Out		In		Out	
	avg	std	avg	std	avg	std	avg	std
FNN	2.04	1.49	2.21	1.73	11.7	25.2	15	45
RNN	0.7	0.62	0.88	0.85	3.2	9.1	4.96	17.3
eQRNN	**0.37**	0.47	**0.53**	**0.77**	**1.77**	7.53	**3.45**	**18.8**

According to the result, the proposed eQRNN shows 47% and 40% lower MAPE on test in and out set compared to the baseline RNN model. In addition, MSE decreased 45% and 30% for each test sets.

5. Conclusions

We proposed novel deep learning model that predicts the simulation results on thermal hydraulic dynamics in order to broaden the spectrums of accident scenarios to be analyzed. Compared to the baseline deep learning model, mean absolute percentage error is decreased 43% in average.

ACKNOWLEDGEMENTS

This work was supported by a Nuclear Research & Development Program grant from the National Research Foundation of Korea (NRF), funded by the Ministry of Science and ICT (NRF 2019M2C9A1055906), and by (NRF 2020M2C9A1061638)

REFERENCES

[1] H. Kim, J. Cho, J. Park, Application of a deep learning technique to the development of a fast accident scenario identifier, IEEE Access 8 (2020) 177363–177373.
[2] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780.
[3] M. Schuster, K. K. Paliwal, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing 45 (1997) 2673–2681.
[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, arXiv preprint arXiv:1706.03762 (2017).