공학박사학위논문

Development and Applications of Hazardous Gas Dispersion Surrogate Model Using Machine Learning

기계 학습을 활용한 유해 가스 확산 대리 모델의 개발과 응용

2018년 8월

서울대학교 대학원

화학생물공학부

전 경 우

Abstract

Development and Applications of Hazardous Gas Dispersion Surrogate Model Using Machine Learning

Kyeongwoo Jeon

School of Chemical & Biological Engineering

The Graduate School of Seoul National University

The method of predicting hazardous gas dispersion using Computational Fluid Dynamics (CFD) is advantageous in that it can be utilized in various ways due to its high accuracy. However, it has structural complexity and requires a lot of computing resources, which hinders its utilization.

First, TOXIM, which is a toxic gas dispersion simulator that can be easily used by anyone, has been developed solving the structural complexity of computational fluid dynamics. The stability of the simulator is confirmed by carrying out case study under various conditions using TOXIM. In addition, TOXIM's methodology of simulating gas leak accident that occurred in Gumi prove similarity between actual and simulation results.

Next, a surrogate model is developed to reduce computation time of

computational fluid dynamics and utilize it in real time. Compressed computational fluid dynamics calculation results are obtained by using various deep learning methods, and a surrogate model is created using neural network based functions. Among them, VAEDC-DNN shows the highest accuracy. It is confirmed that the computation time is reduced to several seconds in the surrogate model compared to several hours in the actual computational fluid dynamics. This demonstrates that computational fluid dynamics can be used in real time in a gas leak accident.

Finally, the CFD model is applied to the gas leak detector placement problem. A surrogate model is used to generate various gas leak accident scenarios and the position of the detector is optimized by applying the Mixed Integer Linear Programming using the generated samples. Optimization results showed better performance when using surrogate model.

Keywords: Computational Fluid Dynamics; Surrogate model; Hazardous gas dispersion; Machine learning, Deep learning

Student ID: 2014-21541

Contents

List of Figures

List of Tables

CHAPTER 1 : Introduction

1.1. Research motivation

In the chemical process industry, predicted potential accident damage is used as an indicator of the safety and risk of the process. It is also possible to establish precautions based on precise predictions of potential accidents. Computational Fluid Dynamics (CFD) is a very powerful tool among many ways to calculate accident damage. CFD enables modeling of various accidents that can occur in the plant, such as hazardous gas dispersion accidents, fire accidents, and explosion accidents. CFD is also more accurate than other 2D models because it can account for the effects of terrain and obstacles within the scope of the accident.

Various software have been developed to apply CFD to accidents simulations. However, structural complexity and excessive computational resource requirements have hampered CFD software becoming the most popular tool. Although computational power has become stronger due to advances in technology, there is still a limit to the use of CFD in emergencies after an accident. It can only be used for accident damage prediction and analysis before an accident occurs. These disadvantages must be solved in order for CFD simulations to be applied to industries in a variety of situations and ways.

1.2. Mathematical formulation of CFD

The FLACS (Flame Acceleration Simulator) used in this thesis solves three dimensional Reynolds-averaged Navier-Stokes (3D RANS) equations, which are widely used in other CFD studies, to calculate compressible fluid flow. This 3D RANS equation is based on the k-ε turbulence model (Launder & Spalding, 1974) in non-uniform Cartesian coordinates. In addition, to effectively calculate obstacles that are smaller than the sub-grid and reduce computational costs, distributed porosity concepts are used (Hjertager, 1986).

In this thesis, two main governing equations are used, one for the mass conservation equation and the other for the momentum conservation equation. Mass conservation equation can be written as:

$$\frac{\partial}{\partial t}(\beta_v \rho) + \frac{\partial}{\partial x_j}(\beta_j \rho u_j) = \frac{\dot{m}}{V} \quad (1)$$

Where subscript β_v is volume porosity, β_j is area porosity in the j direction. And Conservation of momentum equation can be written as:

$$\frac{\partial}{\partial t}(\beta_v \rho u_i) + \frac{\partial}{\partial x_j}(\beta_j \rho u_i u_j) = -\beta_v \frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}(\beta_j \sigma_{ij}) + F_{o,i} + \beta_v F_{w,i} + \beta_v(\rho - \rho_0) g_i \quad (2)$$

Where $F_{o,i}$ means flow resistance created by sub-grid obstacles, $F_{w,i}$ is flow resistance created by walls and σ_{ij} is the stress tensor. Here $F_{o,i}$ and σ_{ij} are given by:

$$F_{o,i} = -\rho \left|\frac{\partial \beta}{\partial x_i}\right| u_i |u_i| \quad (3)$$

$$\sigma_{ij} = \mu_{eff}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\left(\rho k + \mu_{eff}\frac{\partial u_k}{\partial x_k}\right) \quad (4)$$

In equation (4), the effective viscosity (μ_{eff}) is given by:

$$\mu_{eff} = \mu + \rho C_\mu \frac{k^2}{\varepsilon} \quad (5)$$

Here, the second term here is called turbulent viscosity or eddy viscosity, where k means turbulent kinetic energy and ε means dissipation of turbulent kinetic energy.

1.3. Outline of the thesis

Chapter 1: Introduction of thesis

Chapter 2: Development and Applications of Hazardous Gas Dispersion Simulator: TOXIM

Chapter 3: Toxic gas dispersion modeling for real-time analysis using variational autoencoder with convolutional neural networks

Chapter 4: Development of surrogate model using CFD and artificial neural networks to optimize gas detector layout

Chapter 5: Conclusions

1.4. Associated publications

The work presented in Chapter 2.6 is based on (Yang, 2017). More detailed information of the work is not discussed in this thesis. The work presented in Chapter 3 is based on (Na and Jeon, 2018) with J. Na (joint first author). The work presented in Chapter 4 has been submitted to KJCE. The paper is reconstructed and developed based on the contents that was first introduced in Yang`s doctoral thesis.

Chapter 2: Development and Applications of Hazardous Gas Dispersion Simulator: TOXIM

2.1. Chapter outline

In this chapter, development of TOXIM (TOXIC gas risk analyzing sIMulator) is discussed. Part 2.2 covers the development background of TOXIM. Part 2.3 deals with the development process and methodology of TOXIM. Part 2.4 and 2.5 deal with development results and case studies. Finally, part 2.6 discusses validation to verify the accuracy of the simulator.

2.2. Background

As the chemical engineering industry grows steadily, concerns about catastrophic accidents in chemical plants have increased. Among the various types of accidents in chemical processes, toxic gas leaks are particularly critical owing to the wide range of the risks involved. Prediction of gas dispersion in urban areas is important because urban areas can have more casualties in case of an accident.

Among the methods for predicting gas diffusion, Computational Fluid Dynamics (CFD) is a powerful tool because of its high accuracy. Due to the development of the IT industry, computational power has progressed so much that CFD can now be computed in a personal computer. Still, CFD has been regarded only as an area of expertise due to its structural difficulties. For example, gas dispersion simulations in urban areas require modeling of 3D geometry such as buildings and terrain. In addition, the leak rate should be calculated along with the physical properties of the material being leaked. This complexity makes it difficult for workers in related fields to utilize CFD. In order to solve these problems, TOXIM, which is a powerful and simple software that includes the properties of CFD, is developed.

2.3. Methodology

For gas dispersion simulation using CFD, four major inputs are needed: weather data, storage data, material data and geometry data. The direction, velocity, and temperature of the initial wind are necessary to generate the wind field inside the geometry. Also, storage information should be entered to calculate leakage flow and velocity. To calculate the fatality due to an accident, information on the toxicity of the leaked substance is also needed. Finally, geometry data for the building and terrain of the accident area is required. All the data except for geometry consist an initial condition of accident scenario, and CFD calculation is performed in the grid created on the geometry with the initial condition. The CFD engine used in this TOXIM alpha version is FLACS.

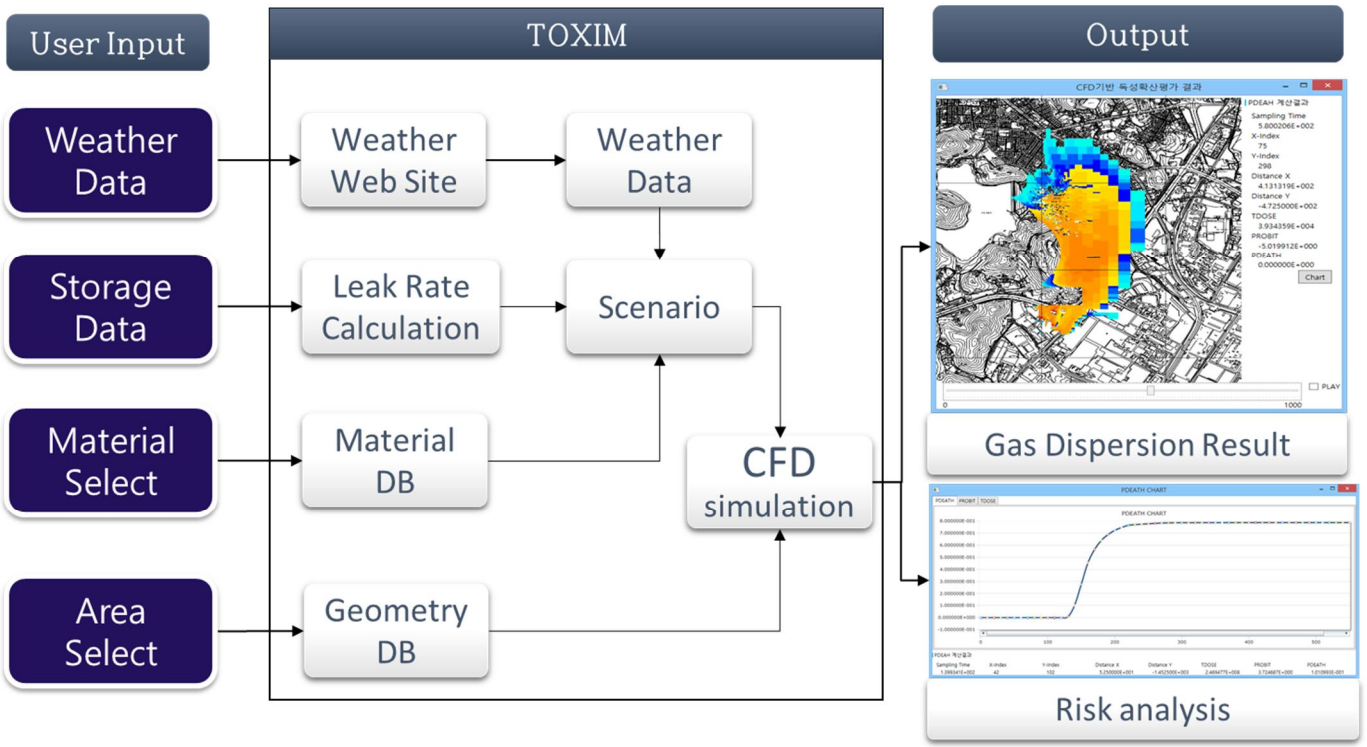

Figure 1 Flow chart of data processing in TOXIM

- Discharge model

To calculate the mass flow rate of the leak, a discharge model should first be made. In TOXIM, simple Bernoulli equation is used.

$$\dot{m}(t) = C_d A\sqrt{2\rho_f(p(t) - p_a)} \quad \ldots \text{(Eq 2-3)}$$

Where, $\dot{m}(t)$: mass flow rate $(\frac{\text{kg}}{\text{s}})$

C_d: coefficient (0.62)

A: leak area (m^2)

ρ_f: density of fluid $(\frac{\text{kg}}{m^3})$

p(t): pressure of the tank (Pa)

p_a: atmospheric pressure (Pa)

- Fatality calculation

There are a number of indicators that can be used to assess the risk of exposure to toxic gases. Among these, AEGLs (Acute Exposure Guideline Levels), ERPGs (Emergency Response Planning Guidelines), and TEELs (Temporary Emergency Exposure Limits) are widely used. Moreover, there is a method from Withers and Lees (1985a, b) to quantitatively calculate the probability of death using the probit function. This method is a vulnerability model for describing the average fatal effects due to chlorine release. The equation of this method can be written as

$$Pr = a + b\ln(\int c^n dt) \quad (6)$$

$$P_{death} = 0.5 + \left(1 + \frac{\mathrm{erf}(Pr-5)}{\sqrt{2}}\right), \quad (7)$$

where Pr is the probit, c is the concentration by volume (in ppm), and a, b, n are constants. In TOXIM, the probit value is integrated up to 10 minutes after the onset of the leak.

2.4. Development result

TOXIM consists of three parts: data input part, calculation process part and result part. The toxic gas dispersion simulation proceeds through these three parts.

In the data input part, basic data required for gas dispersion modeling are input. First, select the type of material that has leaked. When a material is selected, the constants stored in the data base are called. Then, meteorological data should be input. There are manual and automatic modes for entering meteorological information. In manual mode, information such as wind direction, speed, and temperature is input directly. In the automatic mode, weather information of the current area is automatically loaded from the weather station. Finally, the leak point is set by clicking on the map and the leak mass flow is calculated when the whole size, the leak duration, and the internal pressure of the tank are entered.

In the calculation process part, you can check the progress of the calculation, stop the calculation and correct the input information.

In the result part, the concentration of the substance with time can be confirmed by the 2D contour. At each point, the concentration of the substance and lethality can be shown in plot form.

Figure 2 Data input part of TOXIM

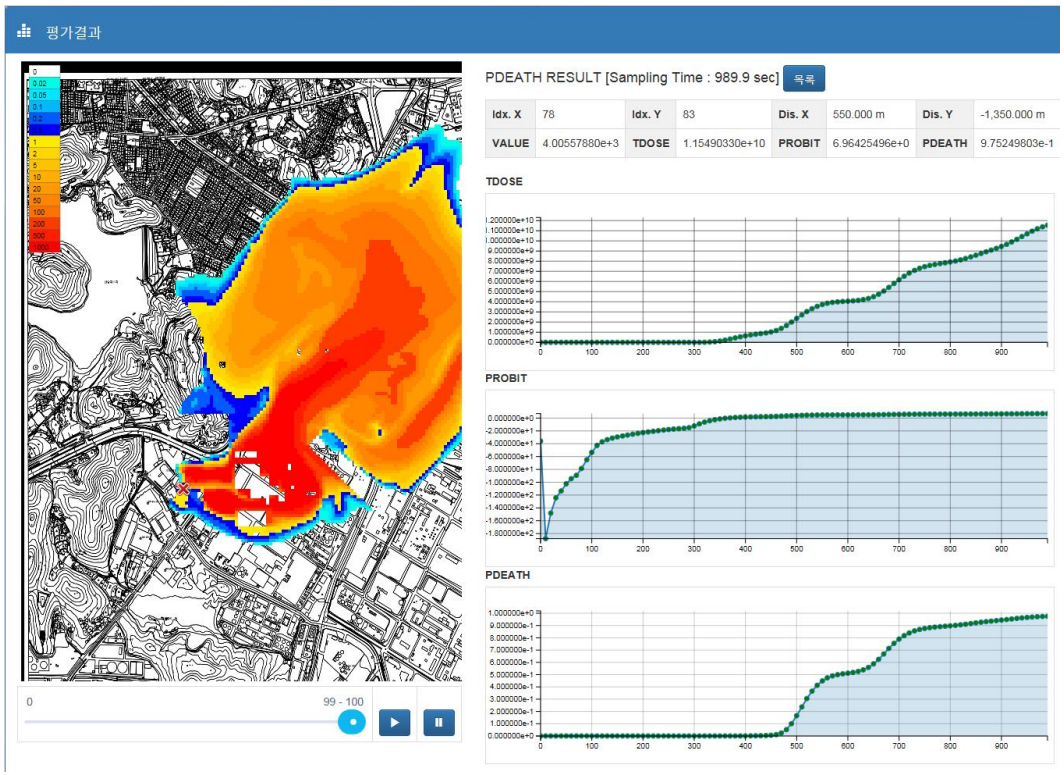

Figure 3 Simulation result part of TOXIM

2.5. Toxic gas dispersion case study using TOXIM

In order to confirm the performance of TOXIM, such as the result, structural stability, and calculation speed, case study is performed in various situations. A chemical plant in the Ulsan area is fixed at the point of leakage and a number of results are obtained by changing the wind direction, wind speed, leaking amount, and material.

- Wind speed change

Figure4, 5 shows the result obtained by changing only wind velocity under the same conditions. The chlorine leaks from the target point with leak rate of 6.6 kg / s for 60 seconds. Wind direction is set from south to north (180°) and wind speed is set at 3m / s and 5m / s, respectively. The results are obtained at 5 minutes and 10 minutes after the start of the leak.

- Wind direction change

Figure6 shows the result obtained by changing only wind direction under the same conditions. The chlorine leaks from the target point with leak rate of 7.4kg/ s for 60 seconds. Wind speed is set to 5m / s and wind direction is set to 150°, 180° and 210°respectively. The results are obtained at 10 minutes after the start of the leak.

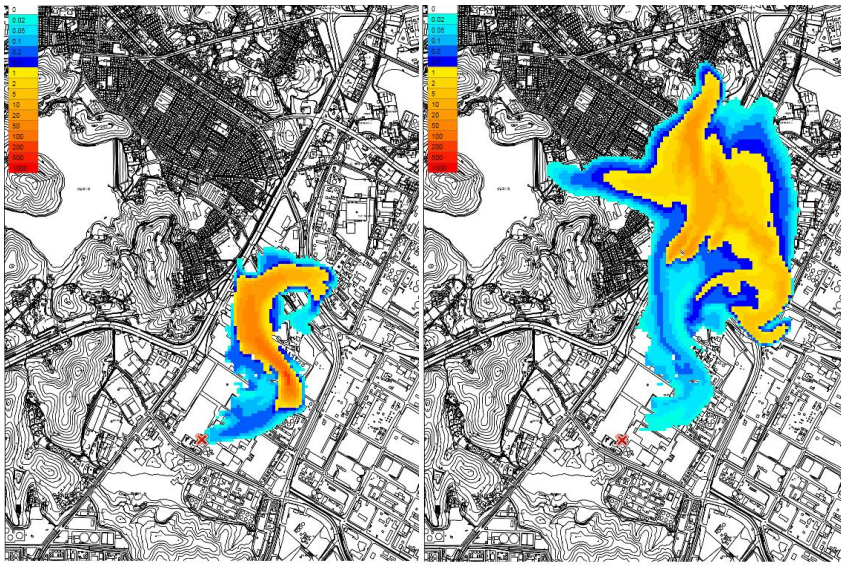

Figure 4 Wind speed change case with 3m/s after 5min(left) and 10min(right)

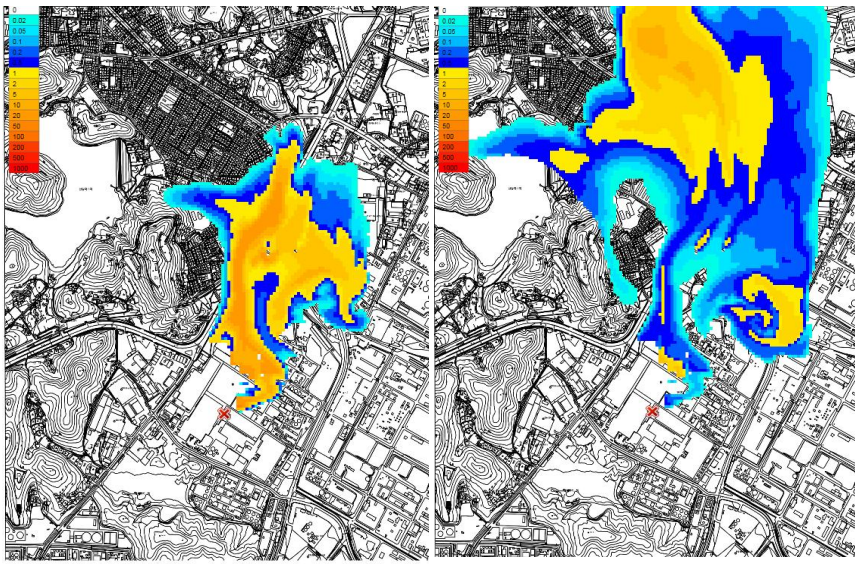

Figure 5 Wind speed change case with 5m/s after 5min(left) and 10min(right)

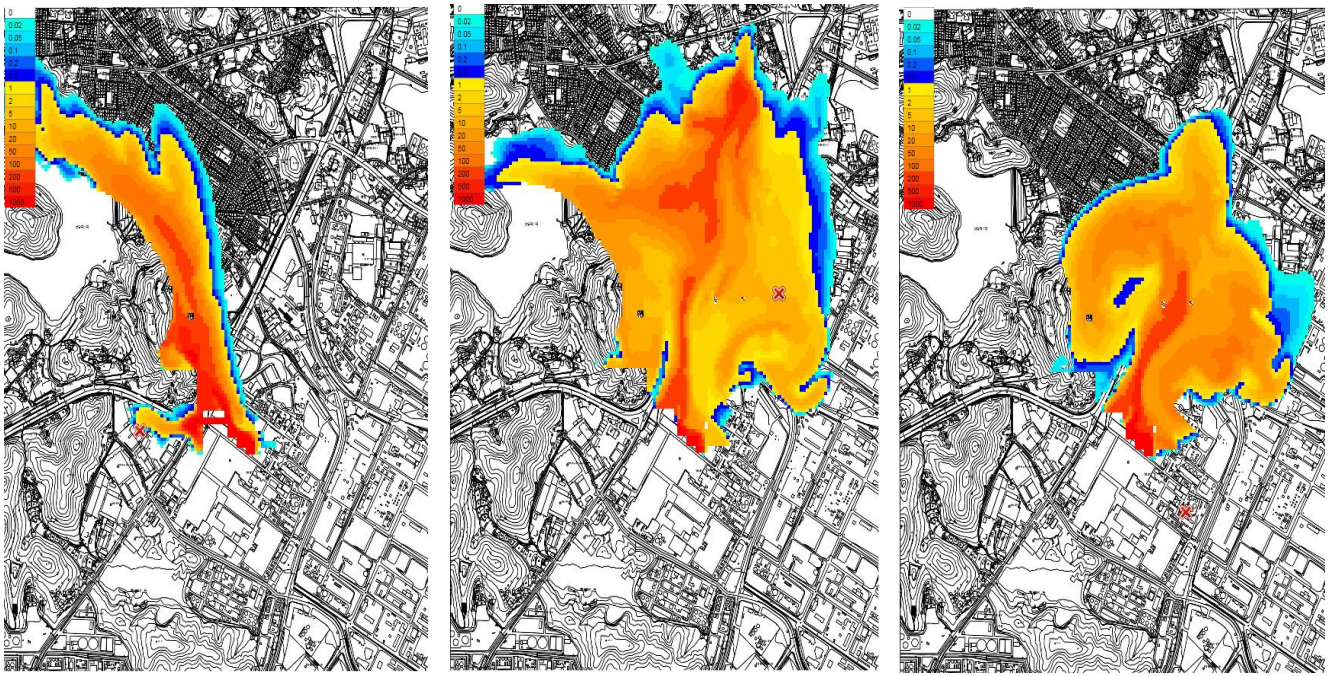

Figure 6 Wind direction change case with 150° (left), 180° (mid) and 210° (right) after 10min

- Material change

Figure7 shows the result obtained by changing only material species under the same conditions. The chlorine and ammonia leaks from the target point with leak rate of 6.6kg/ s for 60 seconds. Wind speed is set to 5m / s and wind direction is set to 180°. The results are obtained at 10 minutes after the start of the leak.

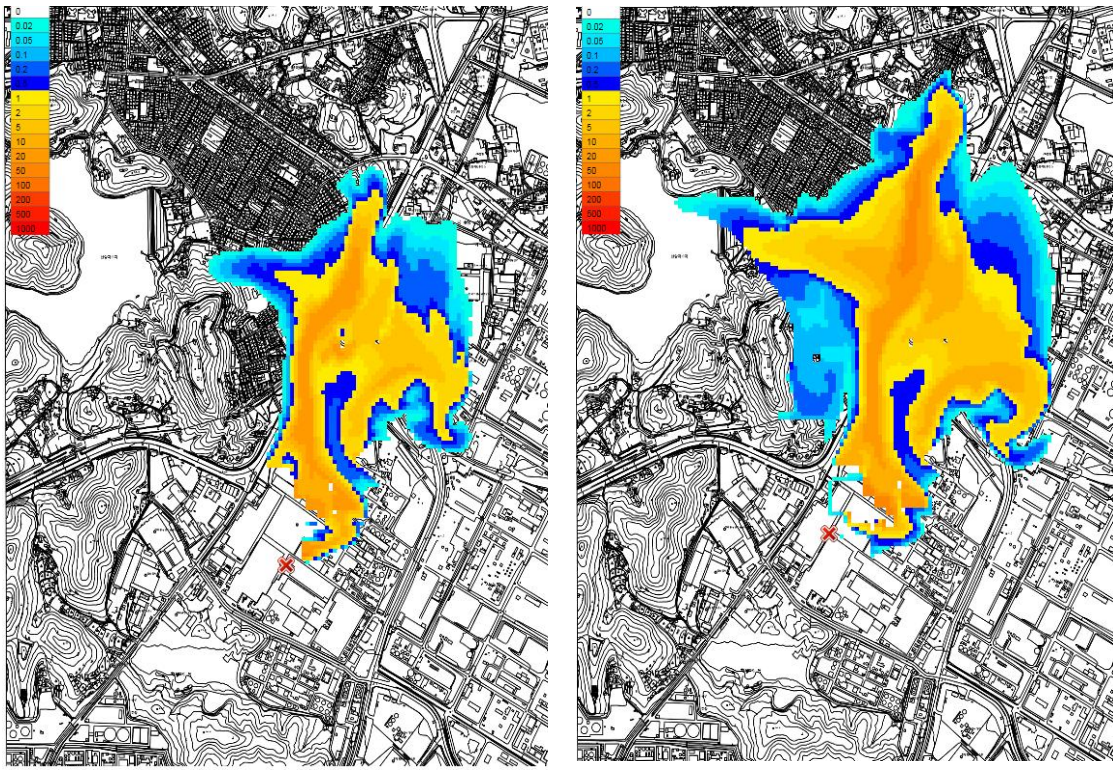

Figure 7 Material change case with chlorine(left) and ammonia(right)

2.6. Validation

In order to confirm the accuracy of TOXIM 's methodology, we modeled and validated HF leak accident in Gumi, Korea (Yang, 2017).

The hydrogen fluoride accident occurred on September 27, 2012 at an LCD cleaning solvent production plant in Gumi, South Korea. The accident killed five people, injured 12, and damaged 2.4 km^2 of an agricultural field, in addition to causing harm to 3200 domestic animals (Korea Occupational Safety & Health Agency 2013). The leak's source was an 18-ton tank container, and CCTV footage from the scene showed a worker accidentally opening the lever. The accident occurred at 15:43 KST, and the valve was completely closed at 23:40. During the 8 hours for which the valve was open, between 8 and 12 tons of hydrogen fluoride gas were released (Lee et al. 2013; Joo 2013).

Since it was difficult to measure the gas dispersion range and concentration at the time of the accident, some researchers tried to estimate the affected area by using fluoride contents in dead vegetation (Gu et al. 2013; Koh 2014; Yim 2016). Figure 8 shows the toxic dose contour simulated by CFD around the leak site. Black dots represent the dead vegetation collection positions. Simulation results show that the dispersion range of gas and the positions of dead vegetation are almost identical.

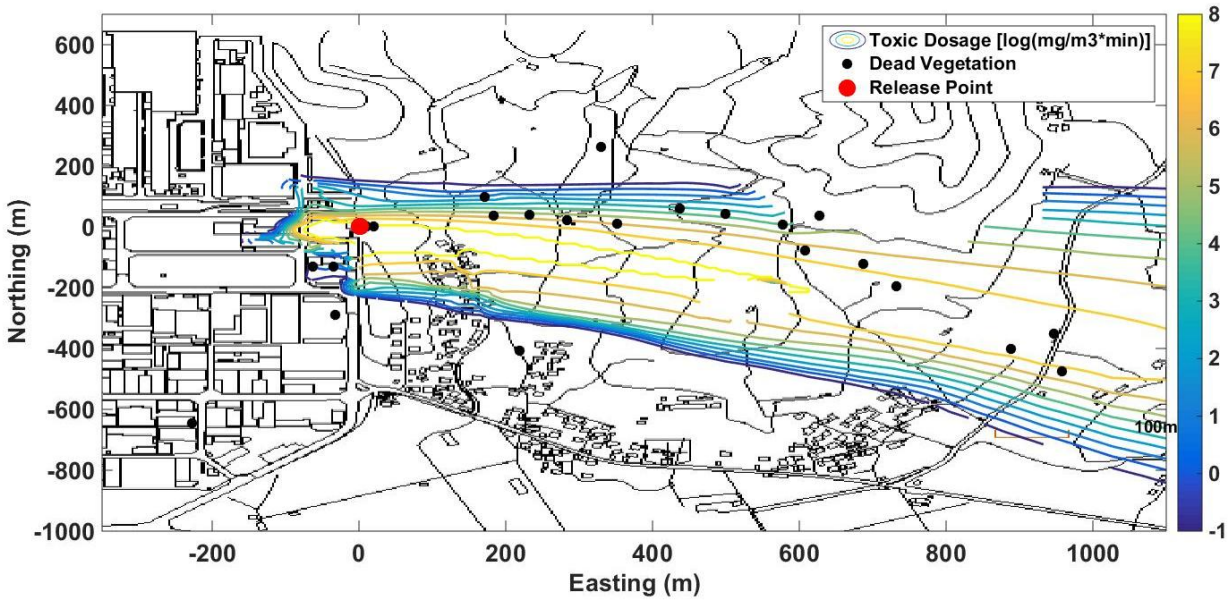

Figure 8 Validation result of toxic dosage

Chapter 3: Toxic gas dispersion modeling for real-time analysis using variational autoencoder with convolutional neural networks

3.1. Chapter outline

High-accuracy gas dispersion models are necessary for predicting toxic gas movement, and for reducing the damage caused by toxic gas release accidents in chemical processes. In urban areas, where obstacles are large and abundant, computational fluid dynamics (CFD) would be the best choice for simulating and analyzing scenarios of accidental release of toxic chemicals. However, owing to the large computation time required for CFD simulation, it is inappropriate in emergency situations and in real-time alarm systems. In this chapter, a non-linear surrogate model based on deep learning is proposed using a variational autoencoder with deep convolutional layers and a deep neural network with batch normalization (VAEDC-DNN) for real-time analysis of the probability of death (Pdeath). VAEDC can extract representation features of the Pdeath contour with complicated urban geometry in the latent space, and DNN maps the variable space into the latent space for the Pdeath image data. The chlorine gas leak accident in the Mipo complex (city of Ulsan, Republic of Korea) is used for verification of the model. The proposed model predicts the Pdeath image within a mean squared

error of 0.00246, and compared with other models, it exhibits superior performance. Furthermore, through the smoothness of image transition in the variable space, it is confirmed that image generation is not overfitting by data memorization.

3.2. Background

As the chemical engineering industry grows steadily, concerns about catastrophic accidents in chemical plants have increased. Among the various types of accidents in chemical processes, toxic gas leaks are particularly critical owing to the wide range of the risks involved. When toxic gas dispersion occurs, a toxic gas cloud is generated and remains in the accident area. If the concentration of the toxic gas is above a certain level, the local population is subjected to serious health hazards. When an accident occurs in a densely populated area, e.g. an urban area, the risk becomes even higher. On December 3, 1984, for example, a methyl isocyanate gas leak accident caused at least 3800 deaths and about 100,000 permanently disabled in Bhopal, India (Broughton, 2005). Therefore, methods for predicting gas dispersion in accidents in urban areas are essential. However, toxic gas dispersion experiments are difficult to implement owing to cost, risk, and technical requirements.

To resolve this, computer simulation has been developed to model gas dispersion. There are two-dimensional integral models, viz. the Box model and Gaussian dispersion models, (Barratt, 2013) and simulators (ALOHA and PHAST), which can be used in emergency owing to their short computation time. They are fairly accurate in the case of flat terrain without three-dimensional obstacles. However, they have low accuracy in the case of three-dimensional obstacles, such as mountains and buildings, as these obstacles are ignored. Moreover, they do not change dynamically, as they are based on the

steady state Thus, it is difficult to use these tools in actual emergency situations (Delaunay, 1996; Hanna and Strimaitis, 1988).

Computational fluid dynamics (CFD) is widely used to accurately simulate the dispersion of toxic chemicals in urban areas. CFD has the highest accuracy because it takes into account geographic information and gas characteristics. The CFD model in urban areas has been developed in various studies. Hanna et al. (2006) described and compared five CFD models using gas dispersion experimental data from Madison Square Garden in Manhattan (MSG05). Hanna et al. (2009) simulated a hypothetical chlorine railcar accident in the Chicago urban area using FLACS and compared the results with those obtained by simpler models, such as SLAB, HGSYSTEM, and ALOHA. Likewise, Long et al. (2009) compared a gas dispersion CFD model (AcuSolve) and the Urban Dispersion Model (HPAC) in regions with large obstacles. Xie et al. (2013) modelled gas flow and dispersion in London using large eddy simulation.

However, CFD-based simulation models have a critical disadvantage, i.e. computation cost. Most accident scenarios require more than 1 h for simulation in CFD-based tools, such as FLACS, Fluent, and OpenFOAM (Hanna et al., 2009; Middha et al., 2010; Yang et al., 2017). When a toxic gas spill occurs, the simulation time must be at least less than the golden time, also known as golden hour, to effectively predict the extent of the accident and evacuate the population. However, there are currently no CFD-based tools for simulating accidents faster than the golden time. Hence, these tools cannot practically be used in real-time alarm systems. Therefore, there is a need for

computationally efficient methods that are as accurate as CFD. Therefore, research on surrogate or meta-models have been conducted to simplify complex models and shorten computation time. Palmer and Realff (2002) and Caballero and Grossmann (2008a, b) used surrogate models based on Gaussian process regression (GPR), also known as kriging model to optimize flowsheet simulation. Gomes et al. (2008) used GPR for real-time process optimization. Chen et al. (2011) applied meta-models of complex process simulations with time-space-dependent output adopting GPR. Moreover, Wang et al. (2014) achieved data reduction using segPCT-PCA and GPR meta-models. Loy et al. (2017) developed and compared two surrogate models for consequence analysis based on CFD, i.e. a non-linear global surrogate model (least squares support vector machine) and a linear piece-wise surrogate model (linear nearest neighbor interpolation). Research for developing a meta-model has been steadily progressing, and methods such as GPR and support vector machines have been primarily used. In addition, Kajero et al. (2017) has reviewed other methods for this meta-modeling.

Three steps are required for designing a surrogate model based on large amounts of data: data reduction, data regression, and data reconstruction. In most of the above-mentioned studies, since data reduction uses a linear method such as PCA, it is often difficult to distinguish nonlinear manifolds or to store image information with complex nonlinearities. Therefore, a training technique for the representation of an unlabeled data set through an autoencoder has been developed. An autoencoder is a learning system based on artificial neural networks that can efficiently learn and compress input

information. Autoencoders are composed of an encoder (recognition network), a decoder (generative network), and hidden layers (internal representation). Since autoencoders are based on neural networks and can operate in various combinations, they achieve sufficient dimensionality reduction even for data sets with strong nonlinearity. Moreover, when combined with a convolutional neural network (CNN), they can perform powerful feature detection, which is a significant performance improvement (Masci et al., 2011). This is because when CNN's filters are well trained, they can effectively extract features for complex image data. Finally, a variational autoencoder that extends to a generative model based on the latent space generated by features of the training set has been developed as a powerful model that can generate images by extracting definite features and changing their values (Kingma and Welling, 2013). This gives a probability distribution to the deterministic vector of the internal representation, thus extracting the most sensitive features. Therefore, variational autoencoder with deep convolutional layers would be a powerful meta-modeling, with the addition of the regression process that performs the mapping between the important variable and the encoded data or the latent space from the autoencoder.

In regard to regression, various methods are used, such as linear regression and artificial neural networks. Recently, various methods have been introduced to deep neural networks (DNN) and resolved the issues of vanishing gradient, excessively low learning speed, and overfitting the training set (Géron, 2017). For example, initialization methods have been introduced (Glorot and Bengio, 2010; He et al., 2015). Moreover, the ReLU

activation function, which exhibits better performance than the sigmoid activation function, and the Mother Nature activation function appeared in the theory of deep networks. In addition, non-saturating activation functions, such as the leaky ReLU, ELU, and SeLU, were introduced, improving performance. It is obvious that these methods can reduce the vanishing and exploding gradient problems at the beginning of training state; however, these may occur during the learning state. To resolve this, zero-centering and normalization of the input is required before the activation function through the Batch Normalization (BN) method proposed by (Ioffe and Szegedy, 2015) and subsequent scaling and shifting of the results. Moreover, techniques such as momentum optimization, Nesterov Accelerated Gradient, AdaGrad, RMSProp, and Adam optimization have been developed to optimize machine learning. Adam (Adaptive Momentum Estimation) optimization is performed by combining Momentum optimization and the RMSProp algorithm (Kingma and Ba, 2014). Therefore, it is highly likely to develop high-performance surrogate models using this state-of-the-art deep learning technique for data reduction with high quality feature extraction, data regression of highly non-linear manifold, and generation or reconstruction of the image data by means of generative models.

In this chapter, a gas leak model of industrial scale was developed for the Mipo complex in the city of Ulsan, Republic of Korea, using FLACS. Moreover, a surrogate model was constructed for real-time applications, namely, a real-time alarm system within golden time. Initially, for the scenario where three variables (wind speed, wind direction, release rate) were

randomly applied, the data was obtained by CFD and the probability of death factor (Pdeath) was expressed as a 2D image. Subsequently, using a variational autoencoder with deep convolutional layers, the resulting data of dimension 30,400 was compressed into a 32-dimensional latent space. Through the deep neural network architecture designed by the authors, the variable space was mapped into the latent space to predict the 32-dimensional latent space with 3 variables. Finally, a surrogate model was constructed that can generate the predicted contour of Pdeath through three variables only. The effectiveness and applicability of the model was demonstrated by comparison with several other methods of regression. It was shown that the model does not merely memorize the Pdeath contour but rather extracts features and smoothly generates the image.

3.3. Toxic gas dispersion CFD model

In this chapter, FLACS (Flame Acceleration Simulator) is utilized as a gas dispersion CFD tool. The version of FLACS used is v10.4 released in July 27th, 2015 and developed by Gexcon. FLACS was originally developed for explosion simulation. However, FLACS may also simulate gas dispersion and fire models. A number of studies have been conducted to verify the accuracy of the atmospheric dispersion model of FLACS used in this study. Hanna (2009) modeled an actual chlorine leakage accident in an urban area and compared the results with the accident data. In addition, Yang et al. (2017) compared the hydrogen fluoride leakage accidents in the Gumi area in Korea with the FLACS simulation results. In addition, several studies have been carried out to validate gas diffusion experiments performed under various conditions, not actual accidents, with FLACS (Dharmavaram et al., 2005; Hanna et al., 2004; Hansen et al., 2010; Middha et al., 2010).

3.3.1. Model description

The Mipo complex in the city of Ulsan, Republic of Korea, has a large number of industrial plants as well as a residential area nearby. Thus, this region has a high potential of not only gas leakage accidents but also considerable damage in case of such an accident. Therefore, it was selected as a virtual gas leakage accident site. In the model, seen on the left side of Fig. 9, flat terrain with thousands of buildings was assumed. The size of the entire domain is about 4,000 m in the *x*-direction, 3,000 m in the y-direction, and 80 m in *z*-direction (from ground level). The CAD image inside FLACS can be seen on the right side of Fig. 9. As more than 500 simulation data points are required, a uniform grid resolution of 20 m × 20 m × 20 m was used to reduce computation cost.

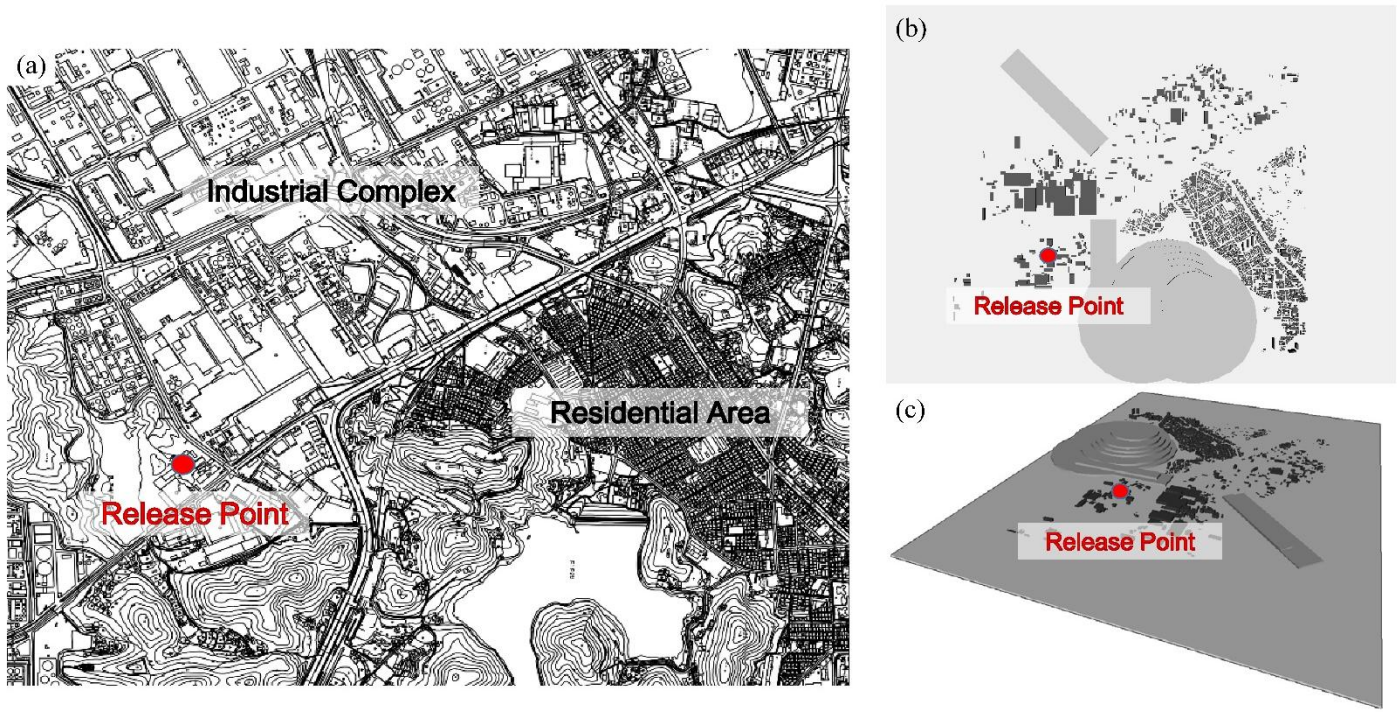

Figure 9 (a) Geometry of Mipo complex in Ulsan, (b) top view of CAD image, and (c) 3D view of CAD image in FLACS.

As liquefied chlorine is used on a large scale in this area for industrial purposes, chlorine was chosen as the material of the hypothetical release scenario. To generate distinct simulation samples, three main variables were randomly selected within a certain boundary. Wind direction was chosen within 0 - 2π, wind speed within 0.5~5 m/s, and release rate within 10~100 kg/s. All other variables were fixed, including release duration (60 s), release point, release material, and temperature (20° C). The entire simulation time in all cases was set to 1000 s. The conditions of the simulation are summarized in Table 1.

Table 1 Scenario conditions

Variable	Unit	Value
Ambient temperature	℃	20
Ambient pressure	Bar	1
Wind direction	rad	0 - 2π
Wind speed	m/s	0.5 – 5
Discharge rate	kg/s	10 – 100
Discharge direction	-	+Y
Release duration	s	60
Total simulation time	s	1000
Pasquill class	-	None

3.3.2. Lethality calculation

Chlorine is a highly toxic material. There are a number of indicators that can be used to assess the risk of exposure to toxic gases. Among these, AEGLs (Acute Exposure Guideline Levels), ERPGs (Emergency Response Planning Guidelines), and TEELs (Temporary Emergency Exposure Limits) are widely used. Moreover, there is a method from Withers and Lees (1985a, b) to quantitatively calculate the probability of death using the probit function. This method is a vulnerability model for describing the average fatal effects due to chlorine release. The equation of this method can be written as

$$Pr = a + b\ln(\int c^n dt) \tag{6}$$

$$P_{death} = 0.5 + \left(1 + \frac{\mathrm{erf}(Pr-5)}{\sqrt{2}}\right), \tag{7}$$

where Pr is the probit, c is the concentration by volume (in ppm), and a, b, n are constants. In the case of chlorine, a is -0.829, b is 0.92, and n is 2 (Perry and Articola, 1980). In this study, the probit value is integrated up to 10 minutes after the onset of the leak. As the toxic gas flows into the residential area for about 10 min during the pre-simulation stage, 10 min is considered the golden time.

3.3.3. Numerical setup

FLACS solves the conservation equations for each cell of a 3D Cartesian grid using the finite volume method. The numerical time step algorithm that is used in FLACS is based on the implicit first-order backward Euler method. Time steps in transient simulations should be set in order that the solution evolve smoothly and stably in time. The Courant-Friedrich-Levy (CFL) number provides a solver-specific criterion for the maximum time step that yields a stable solution in the compressible solver of FLACS. Two CFL numbers are used to determine the maximum time steps: CFLV and CFLC. CFLV is based on fluid velocity, whereas CFLC is based on sound velocity. The CFD solver chooses the minimum value between CFLV and CFLC.

$$\Delta t_v = \frac{\mathrm{CFLV}}{\max(\frac{u_i}{\Delta x_i})} \tag{8}$$

$$\Delta t_c = \frac{\mathrm{CFLC}}{\max(\frac{c}{\Delta x_i})} \tag{9}$$

$$\Delta t = \min(\Delta t_v, \Delta t_c), \tag{10}$$

where Δx_i represents the length of the cell in i-direction and Δt denotes the time step. In this study, CFLV is fixed at 1.0 and CFLC is fixed at 10, which are common values in dispersion simulations. Each simulation is performed by parallel computing with 12 CPU threads. The computer used in

this study has a 24-core Intel Xeon E5-2697v2 (2.7 GHz) processor and 256 GB DDR3 RAM.

3.3.4. Data sampling and preprocessing

To create a surrogate model using a neural network, hundreds of sample data points are required for training. However, it is costly to manually acquire large amounts of data; thus, an automated process is used to generate them. This is carried out by linking Python code with FLACS. Moreover, it is difficult to directly use the results obtained from FLACS in surrogate model training. Therefore, it is necessary to preprocess them. The flow chart for this process can be seen in Fig. 10.

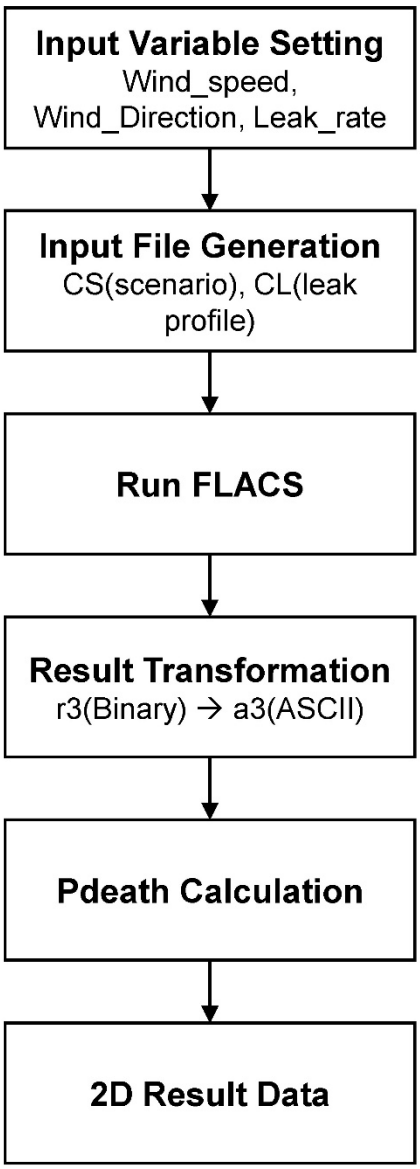

Figure 10 Data sampling and preprocessing flow chart

Initially, among the input variables applied to each sample, three variables (wind speed, wind direction, and release rate) are randomly selected within a certain range. Using these selected variables, the cs and cl files are created. To initiate the FLACS calculation, the cs (scenario), cl (leak), co (geometry), cg (grid), and cp (porosity) files are required. Among them, the co, cg, and cp files are fixed because the simulation is performed with a single geometry. As the variables in the cs and cl files only are changed, they are generated for each sample. Once the FLACS simulation is complete, the resulting r3 file (binary form) can be converted to an a3 file (ASCII form) using the FLACS utility. The result thus obtained is four-dimensional data (x, y, z, and time). Subsequently, Pdeath is calculated by integrating over time up to 10 min. Only the data at the point z = 2 m, which is the height directly affecting the population, is extracted from the reduced 3D data and becomes 2D data (150 × 200), which is the Pdeath value corresponding to the x and y coordinates. 550 samples are thereby generated.

3.4. VAEDC-DNN surrogate model

Most of previous studies on the reduced-order model for chemical engineering problems, such as reactor modeling, gas dispersion modeling, and process data monitoring, used principal component analysis (PCA) for data reduction and feature extraction (Ha et al., 2016; Jin et al., 2006; Kano et al., 2001; Kresta et al., 1991; Lee et al., 2006; Misra et al., 2002; Qin, 2012). However, as deep autoencoders exhibit superior performance in nonlinear data reduction compared with PCA and shallow autoencoders (Hinton and Salakhutdinov, 2006), well configured deep autoencoder are be employed in the present case. Even though PCA can effectively reduce linearly correlated data in certain cases, and there are several applications of PCA, such as kernel-PCA for nonlinear dimensionality reduction or manifold learning, an autoencoder is the nonlinear generalized version of PCA. Thus, keeping the advantages of PCA, it is possible to increase the performance of data reduction by using autoencoder. Furthermore, autoencoders can be applied to the generative model owing to their theoretical relationship with the latent variable model (Goodfellow et al., 2016). That means not only deterministic and rigorous data, but also abstract and stochastic data can be understood and which features can be extracted efficiently. P_{death} images generated from CFD gas dispersion models are highly nonlinear owing to complicated geometric factors (buildings, mountains, and loads), and it is considerably difficult to extract features from them. Thus, a surrogate model based on a variational

autoencoder (Kingma and Welling, 2013) is introduced for compressing the output image data $x \in \mathbb{R}^{152\times200}$ to the latent space $z \in \mathbb{R}^{1\times N_z}$, where N_z is the number of latent variables, using a probabilistic encoder ($q_\phi(z|x)$) and a probabilistic decoder as generator ($p_\theta(x|z)$) by means of variational Bayes that let the prior over the latent variables be the centered isotropic multivariate Gaussian ($\mathcal{N}(z;0,1)$). Finally, the latent space z is mapped by the variable space $v \in \mathbb{R}^{1\times3}$ for reconstructing the image with v only.

For developing the surrogate model ($f^*(v)$) of the CFD gas release model ($f(v)$) employed by FLACS, a variational autoencoder with deep convolutional layer (VAEDC) and a fully connected deep neural network (DNN) involving batch normalization layers are used. The input of the original CFD model and the VAEDC-DNN surrogate model is represented by $v \in \mathbb{R}^{1\times3}$, that is, wind velocity (m/s), wind direction (rad), and gas release rate (kg/s). The output of the models is represented by $x \in \mathbb{R}^{152\times200}$, which is the contour image of P_{death}. Thus, the problem can be defined as developing a high performance, minimum mean squared error, surrogate model $f^*(v)\colon \mathbb{R}^{1\times3} \rightarrow \mathbb{R}^{152\times200}$ of the original CFD model $f(v)\colon \mathbb{R}^{1\times3} \rightarrow \mathbb{R}^{152\times200}$ using the pre-calculated data set of $f(v)$. 500 data points are used as training set ($v^{\text{train}}, x^{\text{train}}$) and validation set ($v^{\text{validation}}, x^{\text{validation}}$). 400 samples were randomly extracted from 500 samples and used for training set. 100 samples were extracted and used for validation set. 50 data points are used for testing ($v^{\text{test}}, x^{\text{test}}$) the final metric (mean squared error) between the generated image data of ($x_{\text{gen}}^{\text{test}}$) and x^{test}.

3.4.1. Model architecture

The proposed surrogate model, VAEDC-DNN, consists of two parts. The first part constructs the encoder and decoder using a variational autoencoder with deep convolutional layers (VAEDC). The second part maps the variable space (v) to the latent space (z) using deep neural networks. This two-stage method enables the proposed model to efficiently regress the P_{death} image (x) with the variable (v).

The detailed VAEDC architecture of the encoder and decoder for the surrogate model is shown in Fig. 11. Three convolutional layers with a kernel size of (3,3), stride of 1, He initialization, ReLU activation function, and padding of the same dimension are used. After each convolutional layer, a max pooling layer with (2,2) filters and stride 2 is used for dimension reduction. Subsequently, the input layer dimension (152 × 200) is reduced to 19 × 25, which is one fourth of the width and height of the image. Then, the 19 × 25 × 8 (number of filters) layer is reshaped into 1 × 3800 vectors for constructing a 3800-128-64 fully connected dense layer with the ReLU activation function. The μ and $\log(\sigma^2)$ layers in parallel represent the mean and log of variance, respectively, of the latent variables. Thus, the dimension of the μ layer, $\log(\sigma^2)$ layer, and z layer is 1 × N_z, where N_z is the number of the latent variables. After the encoding process is finished, the decoding process is started, which is the inverse of the encoding process. Finally, the loss function, which is represented by the variational lower bound, is

calculated using x^{train} and $x_{\mathrm{gen}}^{\mathrm{train}}$ at the variational layer.

The detailed DNN configuration for mapping the latent space from VAEDC and the variable space (wind velocity, wind direction, and gas release rate) is shown in Fig. 12. A 5-layer fully connected deep neural network with batch normalization, ReLU activation function, and He initialization is used for mapping the variable space into the latent space. In the training phase, the network is trained using 400 resampled training sets ($x^{\mathrm{train}'}, v^{\mathrm{train}'}$) and 100 validation sets ($x^{\mathrm{validation}'}, v^{validation'}$). The mean of the latent variables designated as output in this supervised learning is extracted using the encoder part of the VAEDC that has already been trained. The variable set $v^{\mathrm{train}'}$, mapped in one-to-one correspondence, is used as input. Finally, in the generating phase, the model uses a test set that was not used as training or validation set in both VAEDC and DNN. z^{test} is predicted by v^{test} and the trained DNN, and $x_{\mathrm{gen}}^{\mathrm{test}}$ is generated by the decoder of the trained VAEDC. Performance can be assessed by comparing the mean squared errors between $x_{\mathrm{gen}}^{\mathrm{test}}$ and x^{test}.

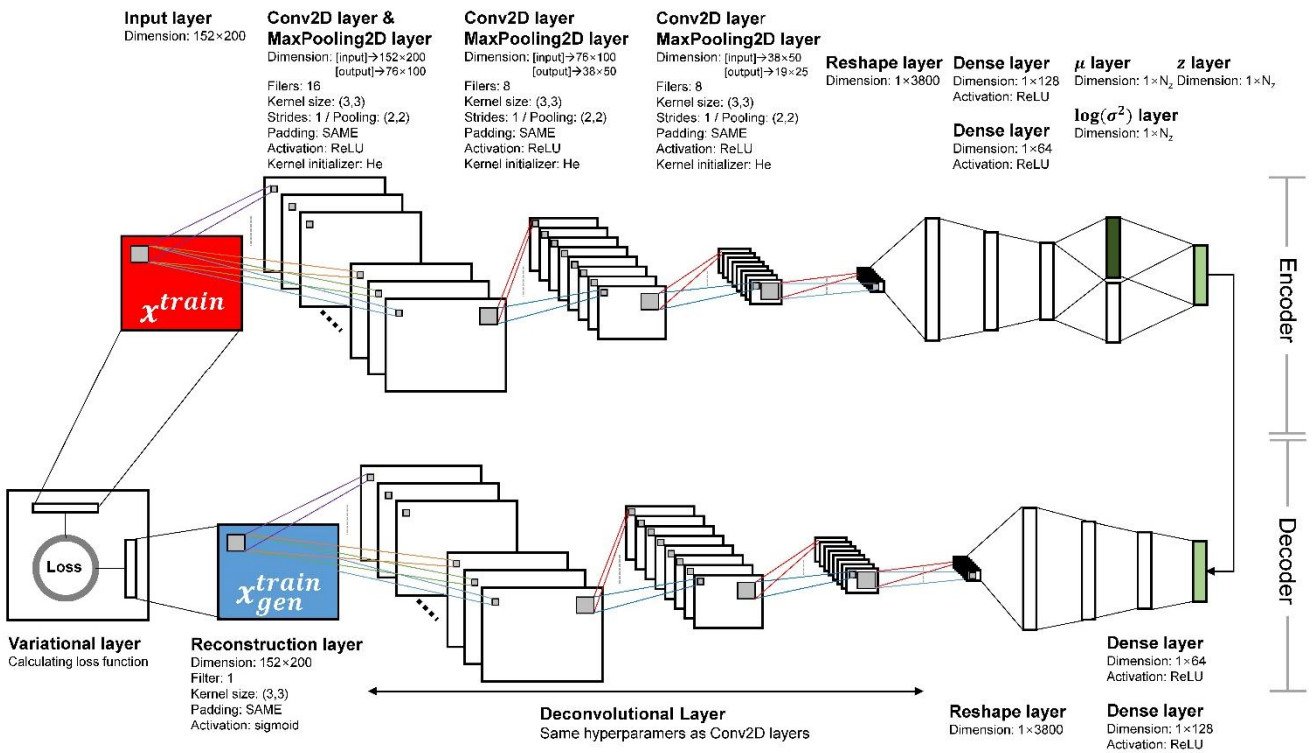

Figure 11 VAEDC architecture

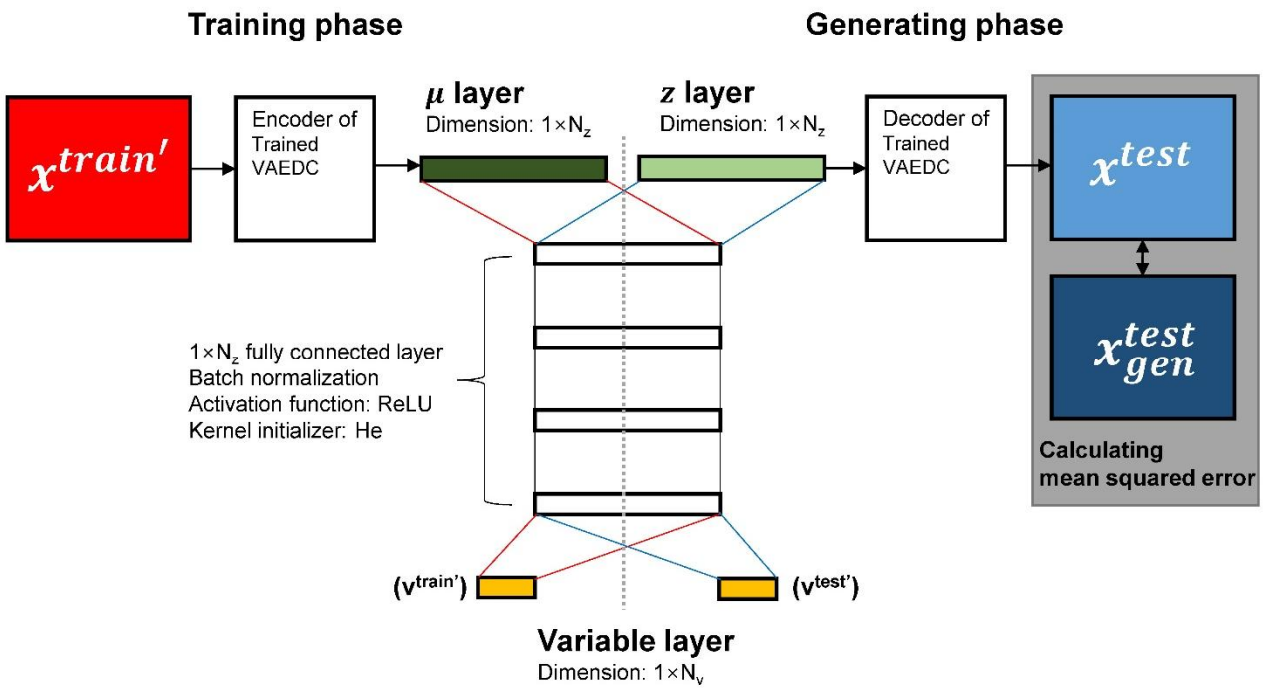

Figure 12 DNN architecture for combining with VAEDC

3.4.2. Latent space and variational lower bound estimator

The loss function is one of the most important settings for constructing efficient autoencoders. In this study, the variational lower bound on the marginal likelihood proposed by (Kingma and Welling, 2013) is used. The lower bound is given by the following equation:

$$\mathcal{L}(\theta, \phi; \mathrm{x}^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x^{(i)}|z)], \quad (1)$$

where D_{KL} denotes Kullback–Leibler divergence, $p_\theta(x|z)$ is the probabilistic decoder with generative parameter θ, and $q_\phi(z|x)$ is the probabilistic encoder with variational parameter ϕ. Training consists in maximizing the variational lower bound on the marginal likelihood of the data point *i* through the variational parameter ϕ and generative parameter θ. To use this statistical approach in a neural network framework, the loss function is defined as $l(\theta, \phi; x^{(i)}) = -\mathcal{L}(\theta, \phi; x^{(i)})$ for converting the maximization problem to a minimization problem. Furthermore, to generate samples from $q_\phi(z|x)$ and reparametrize ϕ as a multivariate Gaussian with diagonal covariance structure with mean (μ) and standard deviation (σ) of the approximate posterior, the following equation is applied to sampling.

$$\mathrm{z}^{(i)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon \text{ and } \epsilon \sim \mathcal{N}(0,1). \quad (2)$$

This sampling is performed at the z layer which is located after the μ and $\log(\sigma^2)$ layers in Fig. 11. Previous studies (Kingma and Welling, 2013) proved that the KL divergence can be calculated and differentiated without

estimation, and the second term of the variational lower bound can be treated as a binary crossentropy loss function. Finally, the proposed loss function for VAEDC and the data point $x^{(i)}$ is given by

$$l(\theta, \phi; x^{(i)}) \simeq -\frac{0.5}{N_z}\sum_{j=1}^{N_z}(1 + \log\left(\left(\sigma_j^{(i)}\right)^2\right) - \left(\mu_j^{(i)}\right)^2 - \left(\sigma_j^{(i)}\right)^2) + \frac{152\times200}{N_{\text{train}}}\sum_{i=1}^{N_{\text{train}}}(x^{(i)} \cdot -\log p_\theta(x^{(i)}|z) + (1 - x^{(i)}) \cdot -\log\left(1 - p_\theta(x^{(i)}|z)\right)) \quad (3)$$

3.4.3. Performance evaluation and numerical setting

It is very important to evaluate the performance of the proposed surrogate model correctly. Evaluating the performance using the value of the loss function of VAEDC or DNN is inappropriate. The purpose of the surrogate model is to predict the P_{death} image correctly using the variable space, rather than merely encode and decode the image data or map the variable space into the latent space. Therefore, it is necessary to determine whether the latent space accurately reflects the features representing the P_{death} image, and whether the variable space and the generated image data are in one-to-one correspondence. That is, it should be ensured that the image data is not simply memorized in the latent space or overfitting. Thus, the performance of the surrogate model is not evaluated by each loss function of VAEDC and DNN but rather using the mean squared error between $x_{\text{gen}}^{\text{test}}$ and x^{test}, where the test set was not used as training or validation set in both VAEDC and DNN. Furthermore, for comparing the model performance with other types of surrogate models based on autoencoder neural networks, a neural network with one hidden layer (NN) without autoencoder, a deep neural network with three hidden layers and batch normalization (DNN) without autoencoder, NN with simple autoencoder (AE-NN), NN and DNN with deep autoencoder (DAE-NN, DAE-DNN), NN and DNN with deep convolutional autoencoder (DCAE-NN, DCAE-DNN), DNN with simple variational autoencoder (VAE-

DNN), and DNN with variational autoencdoer with deep convolutional layers (VAEDC-DNN) are compared using the same hyperparameters (**오류! 참조 원본을 찾을 수 없습니다.**). Detailed architectures and summaries are provided in the supplementary file.

To compare the models fairly, the number of epochs of both the autoencoder and neural network was set to 1,500, and the size of the latent space (N_z) was set to 32 for all models. The size of the variable space (N_v) was set to 3, i.e., wind velocity (m/s), wind direction (rad), and release rate (kg/s). The *Adam* optimizer (Kingma and Ba, 2014) was used for optimizing loss functions. All models were coded using the Python deep learning library Keras with the Tensorflow backend. Training was carried out using NVDIA GeForce GTX 1070 with 8 GB GDDR5.

Table 2 Summary of various models for comparison (architecture, loss function, number of parameters); detailed descriptions are shown in the supplementary file.

		NN	DNN	AE-NN	DAE-NN	DAE-DNN	DCAE-NN	DCAE-DNN	VAE-DNN	VAEDC-DNN
Autoencoder	Description	-	-	One fully connected dense layer	5 fully connected hidden layers		5 fully connected hidden layers, 7 2D convolutional layers, 3 max pooling layers, and 3 up-sampling layers		6 dense layers and a variational layer	9 fully connected hidden layers, 7 2D convolutional layers, 3 max pooling layers, 3 up-sampling layers, and a variational layers
	Loss function	-	-	binary cross entropy					- variational lower bound	
	Parameters	-	-	1,976,032	7,833,696		1,001,881		15,620,352	1,003,961
NN regression	Description	One fully connected hidden layer	Three fully connected hidden layers with BN	NN	NN	DNN	NN	DNN	DNN	DNN
	Loss function	mean squared error								
	Parameters	30,434,400	3,933,056	1,184	1,184	3,680	1,184	3,680	3,680	3,680

3.5. Result and discussion

Various training aspects of the models will be examined through the change of each loss function with respect to the epoch (Fig. 13). The maximum epoch for all training is set to 1500. If this number does not fit, the training may be underfitting or overfitting. This can be determined by comparing the loss function of the validation set with the loss function of the test set. As the number 1500 is set through trial and error, the entire training process is considered appropriate for all models. In the case of NN, a significantly unstable training process manifests itself as the epoch rises over 70. The prediction of a higher dimension from a considerably lower dimension in a fully connected dense hidden layer results in an unstable training process because the number of parameters (30,434,400) is excessively large and no additional normalization process is added. However, it may be used for comparison because the process appears to stabilize to some extent as the epoch exceeds 800. Considering DNN, which has three hidden layers and batch normalization applied after each layer to alleviate the vanishing gradient problem of deep networks, the training process is more stable, without overfitting, and exhibits fast convergence. Regarding AE-NN, the autoencoder part and the NN or DNN parts should be considered separately. In the autoencoder training process, represented by the blue line, overfitting occurs slightly after the epoch 100 only in the case of DAE. However, in AE, DCAE, VAE, and VAEDC, the value of the loss function at the end of training is highly satisfactory. NN, represented by the red line, exhibits a highly stable training process unlike NN without autoencoder. Moreover, there is no

difference between the training error and the validation error. Therefore, there is no overfitting. In the case of DNN, there is a slight vibration in the second half. Nevertheless, it appears to be well trained because it exhibits a monotonically decreasing training process, and there is almost no difference between the validation error and the test error. However, the lower validation error compared with the training error is attributed to the fact that the randomly extracted validation sets, $x^{\mathrm{validation}'}$ and $v^{\mathrm{validation}'}$, contain a predictable subset.

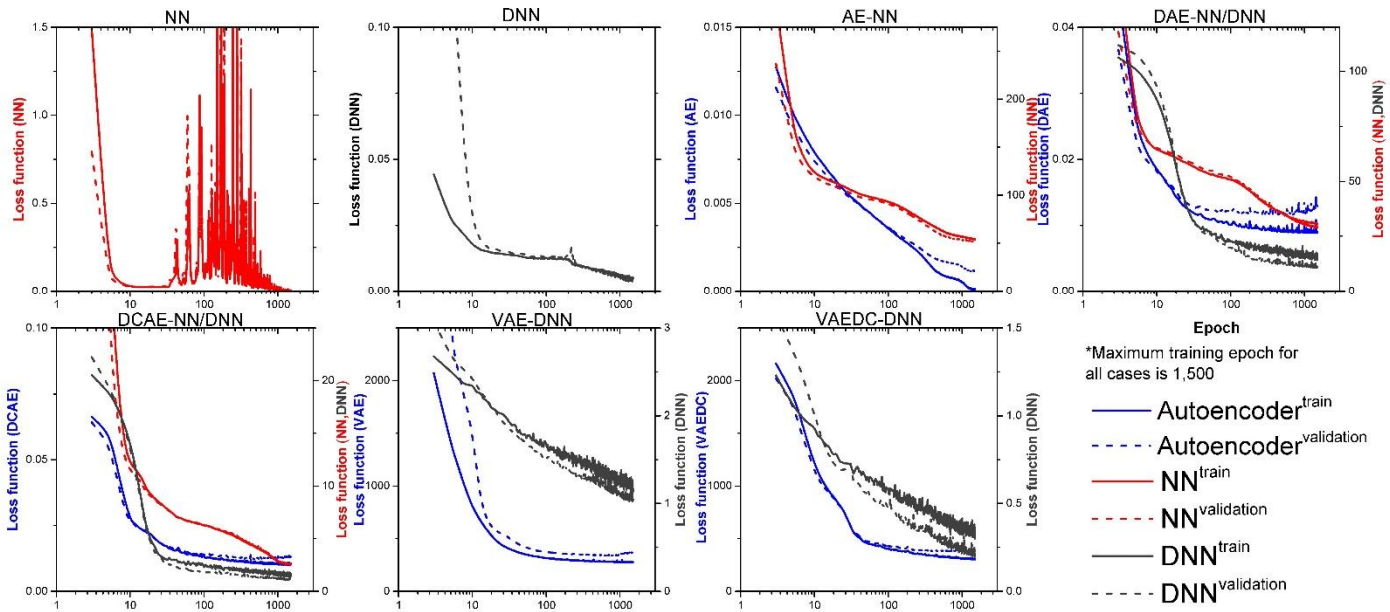

Figure 13 Comparison of epoch vs. loss function graphs for NN, DNN, AE-NN, DAE-NN, DAE-DNN, DCAE-NN, DCAE-DNN, VAE-NN, and VAEDC-DNN models.

In Fig. 14, the mean squared error between $\boldsymbol{x}_{\mathbf{gen}}^{\mathbf{test}}$ and $\boldsymbol{x}^{\mathbf{test}}$ for each of the 50 newly extracted test data sets ($\boldsymbol{x}^{\mathbf{test}}$, $\boldsymbol{v}^{\mathbf{test}}$) is shown. The mean and standard deviation for all test data sets with Gaussian distribution fitting is also provided. The surrogate model constructed using the proposed VAEDC-DNN exhibits the highest performance. The mean value is the lowest, at 0.00246, and the standard deviation of 50 test data sets is considered the narrowest. That is, VAEDC-DNN does not merely memorize the Pdeath image information obtained from the training set, but rather has the ability to extract features in the most suitable form for the latent space and to map it explicitly to the variable space. When NN is used, it can be seen that regardless of the effectiveness of the autoencoder compression, performance drops considerably. It can be seen that the relationship between the latent space and the variable space is not captured clearly. However, performance increases when encoding is such that each variable stored in the latent space clearly exhibits nonlinear correlation with the variable space. It can be confirmed that performance is improved even when NN is used in the order of VAE, DCAE, DAE, and AE. In conclusion, the best performance is achieved when VAE-based autoencoder and convolutional layers are used. In the case of NN without autoencoder, the number of parameters is 30,434,400, whereas in VAEDC-DNN it is 1,007,641, which is 3.31% of the former. The superior performance may be attributed to efficient training and feature extraction with a small number of parameters.

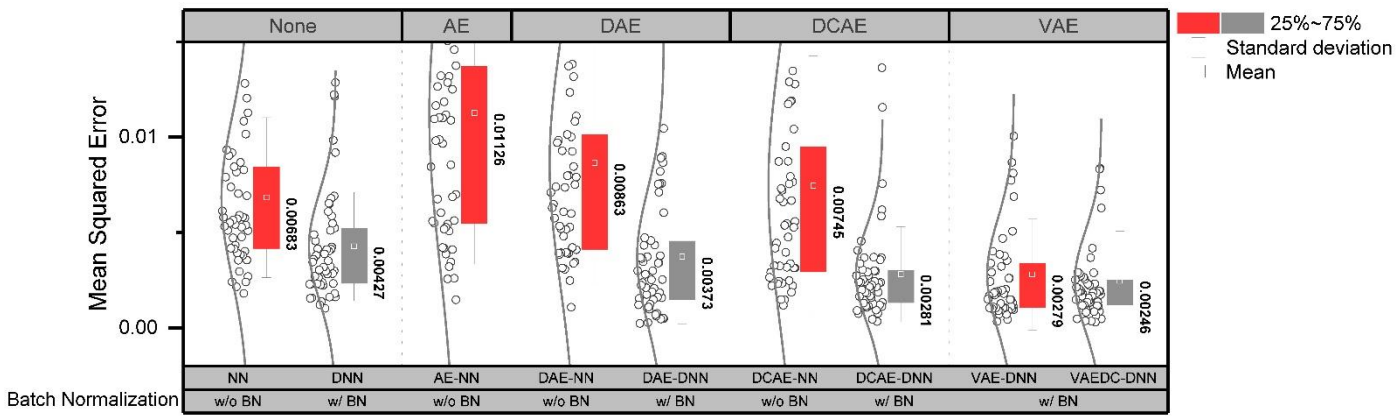

Figure 14 Mean squared error of test set data

It should be noted that in terms of the mean squared error, NN and DNN without autoencoder appear to perform better compared with several combinations of autoencoders and NN. However, if $x_{\mathrm{gen}}^{\mathrm{test}}$ is drawn directly in the contour graph, NN and DNN without autoencoder are shown to exhibit severe noise. To visualize this, a contour graph of five randomly selected $x_{\mathrm{gen}}^{\mathrm{test}}$ generated from FLACS (CFD), which is the ground zero result, NN, AE-NN, DAE-DNN, ACAE-DNN, and VAEDC-DNN are shown in Fig. 15. If NN or DNN without autoencoder are used, severe noise appears. This is because it is difficult to efficiently extract only the features of the image during the training process. Hence, other training data images are memorized and reflected in the results. Moreover, owing to the characteristics of highly nonlinear images, it is difficult to predict the entire image using the variable space only. Thus, the result is quite misleading in some cases, such as the 1st, 3rd, and 5th columns in Fig. 15. The noise disappears from the prediction after data compression by the autoencoder. However, failure to extract features by the autoencoder and map the variable space into the latent space by NN results in deterioration of the surrogate model performance. Such phenomena can be observed directly in AE-NN. The disappearance of noise is important; however, only the approximate trends are predictable. In the case of the 2nd column data, prediction is considerably inaccurate, whereas in the 5th column data it is highly accurate. At this point, it becomes apparent that the introduction of the DNN model with batch normalization model is necessary for improving overall performance. Feature extraction should be activated through the deep convolutional layer and image compression should be

accurately performed with the latent space by predicting the posterior distribution through the variational autoencoder. Between the two best-performing models, i.e., VAEDC-DNN and DCAE-DNN, VAEDC-DNN provides superior prediction compared with FLACS (CFD). In the 1st column, P_{death} spreads out slightly to the upper side; it cannot be detected by DCAE-DNN, whereas VAEDC-DNN detects it. In the 3rd column, VAEDC-DNN is nearly matched although the other models cannot clearly predict width and size. In 4th and 5th column data, superior feature extraction is exhibited even though the shape is complex with many cracking features. Finally, it can be seen that a cracked, non-convex, complex image is predicted as well.

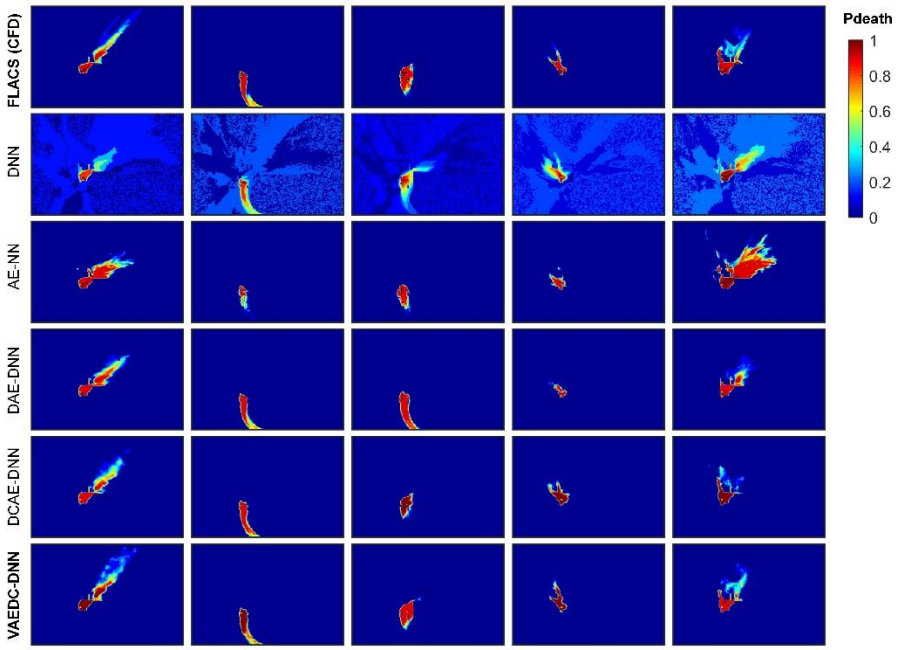

Figure 15 Comparison of generated Pdeath image (xgentest) using vtest with FLACS (CFD), which is ground truth, DNN, AE-NN, DAE-DNN, DCAE-DNN, and VAEDC-DNN.

As can be seen by the sentence “walking in the latent space” from (Radford et al., 2015), understanding the landscape of the latent space is highly important because it is possible to detect memorization through walking in the latent space. If there is sharp image transition, which implies that the latent space is collapsed, the model fails to learn relevant and interesting representations. In this study, as the variable space has already been mapped into the latent space, “walking in the variable space” is suitable for determining whether the manifold is learnt. The results are shown in Fig. 16.

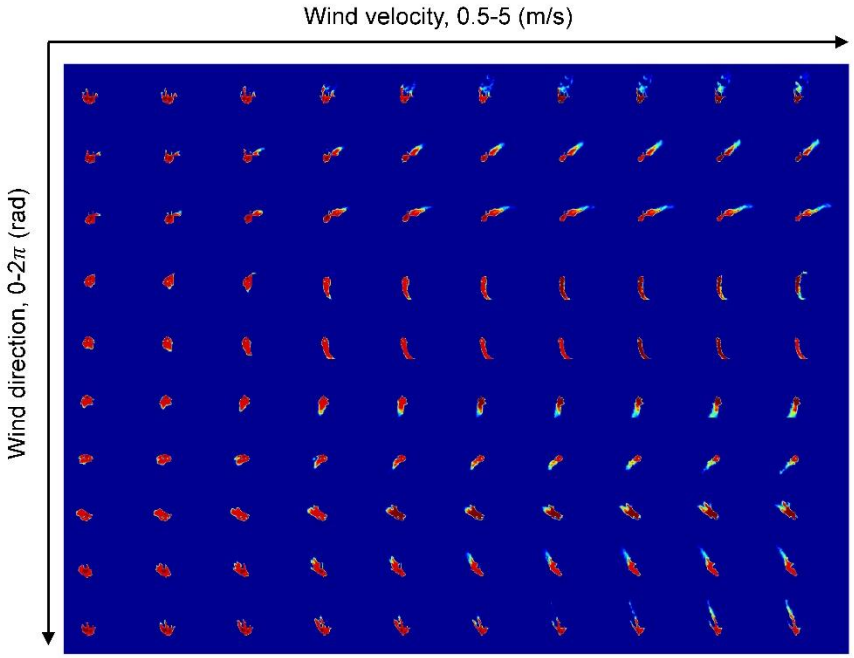

Figure 16 “Walking in the variable space” of a VAEDC-DNN.

As the variable space (v) is three-dimensional, the toxic gas release rate (discharge rate) is fixed at 50 kg/s and P_{death} is visualized as a linear series of 10 points in wind velocity and wind direction. Smooth transitions in all directions are observed. P_{death} spreads wider as wind velocity increases, and the diffusion direction of P_{death} varies smoothly with wind direction, and the topography and obstacles of the region are reflected by the asymmetrical movement. Fig. 16 shows that VAEDC-DNN accurately extracts the P_{death} features, as it is trained not to merely interpolate and symmetrically rotate the wind direction.

Comparison of computational time, storage space for saving the model, and use in real-time alarm systems between the CFD model and VAEDC-DNN is given in Table 3. In the case of real-time alarm systems, if a chemical release accident occurs, then wind velocity, wind direction, and release rate are provided to the alarm system from external sensors. Subsequently, P_{death} or concentration of toxic chemicals should be calculated. When using the CFD model for calculating P_{death}, CPU time is over 700 s using 16 cores of the Intel Xeon E5-2667v3 (3.2 GHz) processor; thus, it is impossible to obtain this information before golden time (10 min). Moreover, it is difficult to pre-calculate all scenarios in advance because the storage requirement for saving CFD data is on the GB scale per case, and results are usually non-linear via the variable space, owing to the complexity of the geometry. Thus, VAEDC-DNN is particularly helpful for developing the surrogate model with a minimum amount of pre-calculated CFD data. After training, up to 1 s of CPU

time with single core is required. As mentioned above, the non-linear correlation between the P_{death} distribution and the geometry effect via the variable space is achieved by the variational autoencoder. Hence, with hundreds of pre-calculated CFD results, VAEDC-DNN successfully predicts the P_{death} contour via any variable space.

Table 3 CPU computational time, storage space for saving the model, and use in real-time alarm systems.

	CPU time	Storage	Use in real-time alarm systems
CFD model	~700 s with 16 cores	GB scale	Thousands of cases are required for predicting all variables.
VAEDC-DNN	< 1 s with single core	KB scale	With hundreds of cases, the surrogate model can predict all variables.

3.6. Chapter Conclusions

In this study, the CFD model was reduced and regression was introduced through a surrogate model for fast calculation. A variational autoencoder with deep convolutional layers was used to extract only key features without noise from high-dimensional image data. Thereby, the CFD result data was reduced, and features were extracted into the latent space z. Moreover, a surrogate model for mapping the variable space v to z using a deep neural network with batch normalization was designed. The integrated model, a variational autoencoder with deep convolution layers interconnected with a deep neural network (VAEDC-DNN), was finally proposed, and the results were remarkable. To verify the performance of the proposed surrogate model, a toxic gas release scenario in the Mipo complex in the city of Ulsan, Republic of Korea, modeled by the commercial CFD software application FLACS was used. CFD modeling was based on CAD, reflecting the complex geometry of real industrial complexes, and the distribution of the probability of death (P_{death}) was obtained as a 2D contour graph by varying v. 500 randomly sampled training sets and 100 validation sets were used for training. Furthermore, 50 randomly sampled test sets, which were not used for training, were used for performance evaluation by comparing the results of the ground truth CFD and those of the proposed surrogate model via the mean squared error of P_{death}. For objectivity, NN without autoencoder, DNN without autoencoder, AE-NN, DAE-NN, DAE-DNN, DCAE-NN, DCAE-DNN VAE-DNN, and VAEDC-DNN were compared using the same hyperparameters. The mse of VAEDC-DNN was 0.00246, which is on average 47.7% as low as

that of the other models. Moreover, the model yields a fairly accurate prediction of the nonlinearity of image cracks and topography. Finally, it is confirmed that image generation is not overfitting by data memorization through the smoothness of image transition in the variable space.

The proposed regression methodology has considerable advantages for developing the surrogate model when the dimension of the training data is very large, which may result in problems related to noise or feature extraction. Furthermore, it leads to high quality surrogate models when the computational cost of the original model, such as a CFD-based model, is overly high for real-time analysis. Therefore, when constructing an early warning system for chemical accidents, or when the dynamics of a virtual plant is required at real time in the form of a 2D or 3D contour image of the concentration of chemicals or temperature profile, compressing the pre-calculated CFD results and developing a surrogate model using VAEDC-DNN is expected to have a considerable effect. If the altered geometry need to be modeled via the proposed regression methodology, then another sampling from the original simulation model for training and validation should be performed. In future work, it is expected that state-of-the-art generative models, such as boundary equilibrium generative adversarial networks (BEGAN), will be used for generating images with sharp edges for more complex features. Furthermore, an alternative surrogate model basis, such as a subset technique like ALAMO (Wilson and Sahinidis, 2017) would be capable of producing better performance.

Chapter 4: Development of surrogate model using CFD and artificial neural networks to optimize gas detector layout

4.1. Chapter outline

When an accident occurs in the chemical industry, the most important factor to reduce the damage of an accident is to detect the accident quickly. To reduce detection time of the gas leak, it is necessary to arrange detectors efficiently. Computational fluid dynamics (CFD) is a powerful tool for simulating gas leaks in a complex plant where there are many structures and obstacles. However, the computational cost is too much to generate sufficient scenarios using CFD. In this chapter, a surrogate model is developed using an artificial neural network based on 30 samples generated using FLACS. Based on 130 samples generated from this surrogate model, sensor allocation optimization was performed using MILP. Cross-validation was performed to confirm the accuracy of the surrogate model. The mean squared error value ranged from 0.0411 to 0.0652.

4.2. Introduction

4.2.1. Detector allocation problem

Major accidents occurring in the chemical industry can be classified into several types. When toxic gases such as chlorine, ammonia and hydrofluoric acid are leaked, there is a serious risk of fatalities for people within the threshold concentration range. Likewise, leaks in chemical processes using flammable gases lead to fire accidents such as jet-fire. If congestion is accompanied by the presence of flammable gas, it leads to a large explosion. All of the accidents mentioned above originate from gas leaks and can be prevented from leading to major accidents if early detection and rapid response occurs. Of course, for early detection to occur, detectors with high performance are required. However, installing a large number of detectors is not only costly but also increases the likelihood of fault alarms and lowers reliability. In other words, the most efficient way to achieve early detection is to place the appropriate number of detectors in the optimal position.

For gas detector layout optimization, accurate gas dispersion modeling should be preceded to confirm the gas flow. One of the most accurate gas diffusion modeling methodologies is Computational Fluid Dynamics (CFD). Unlike other 2D-based modeling methods, CFD is more accurate because it reflects the effects of various obstacles. Several previous studies have verified CFD accuracy by comparing gas diffusion experiment data with CFD simulation results(Hanna et al., 2006; Hanna et al., 2009; Long et al., 2009; Xie et al., 2013). Also, there is some research about gas detector layout

problem using CFD. Davis et al.(2015) And Vázquez-Román et al. (2016) solved gas detector layout problem using CFD simulation results based on eight wind direction assuming the worst case scenario.

There are also some studies on sensor allocation optimization. Berry et al. (2006) designed the basic process for optimizing detector allocation in urban water networks. Hamel et al. (2006) optimized the sensor location to detect chemical, biological, or nuclear attacks in urban areas using CFD simulation. Legg et al. (2012) optimized gas detector layout problem using stochastic programming approach and CFD simulation. They suggested the Mixed Integer Linear Programming (MILP) problem formulation about minimization of expected detection time, minimization of expected detection time including a coverage constraint, and placement based on coverage alone, and later conditional-value-at-risk (CVaR) is used for optimal placement of gas sensors (Legg et al., 2013). Benavides-Serrano et al. (2013) extended the previous Legg`s work with MILP formulation, unavailability and voting strategies. Benavides-Serrano developed the gas detector allocation optimization Quantitative Assessment (Benavides-Serrano et al., 2015), and AP-median formulation (SPqt) (Benavides-Serrano et al., 2016). The research calculated about 270 leak samples with FLACS, a CFD software, which have 994 potential sensor placement in the 3D geometry (50m, 70m, 20m). Gomes et al. (2014) solved a similar problem using 32 samples.

One of the most influential factors in the solution of optimization is the number of different scenarios. If the number of scenarios is less than the number of potential sensors, the sensors tend to be located only near the

source point, which cannot guarantee good optimization. Therefore, a lot of CFD simulation results should be used as a sample for optimization, but the realistic computational cost is considerable for generating such a large number of samples. To solve this problem, the surrogate model is used to increase the number of samples and reduce the computational cost.

4.2.2. Surrogate model with CFD

As the machine learning technology is developed, the surrogate model enables computer-aided simulation such as CFD. A variety of studies to combine surrogate models with CFD have been conducted in a variety of fields such as mechanical engineering and aerospace simulation. In the chemical safety field, Wang et al. (2014) made a surrogate model of LNG gas dispersion using Gaussian process regression (GPR) and segmented principal component transform-principal component analysis (SegPCT-PCA) with samples made from FLACS and Hamersley sampling method. And Margheri and Sagaut (2016) made a toxic gas dispersion model in the urban area using CFD simulation for uncertainty analysis with c-APK technology (anchored-ANOVA-POD/Kriging). These two studies have demonstrated that it is possible to combine the surrogate model with CFD in the field of chemical safety. Loy et al. (2017) developed two surrogate models, one is a nonlinear global surrogate model (least squares support vector machines), and the other is a linear piece-wise surrogate model (linear nearest neighbor interpolation). Both are surrogate models developed to enhance the CFD-based consequence model. In this paper, this methodology is applied to detector layout optimization which is one of the chemical safety fields.

4.3. Method

The main idea of this paper is that CFD based surrogate models results generate different results of CFD and apply them to the detector allocation optimization problem in a chemical plant. The main factor for avoiding large computational costs is to reduce the amount of simulation.

First, Design of Experiment (DoE) should be done for the surrogate model. As a sampling method, Latin Hypercube Sampling (LHS) is used, which is a widely used method for metamodeling techniques to obtain samples evenly. The Artificial Neural Network (ANN) regression and back propagation are used as a surrogate model fitting method.

Figure 17 summarizes the whole process of this chapter. First, M_1 scenarios with five different inputs are generated as base scenarios using the LHS. Then, to find the detection time of gas detectors, all the base scenarios are solved by CFD. The CFD results are used to construct a surrogate model with ANN. After the model is cross-validated, it is used to model M_2 extended scenarios. Total $(M_1 + M_2)$ scenarios are used to solve MILP and then optimal sensor position is determined. In this paper, $M_1 = 30$ and $M_2 = 100$ is used to avoid situations where the number of potential detectors is greater than the number of leakage scenarios while maintaining a realistic CFD simulation number.

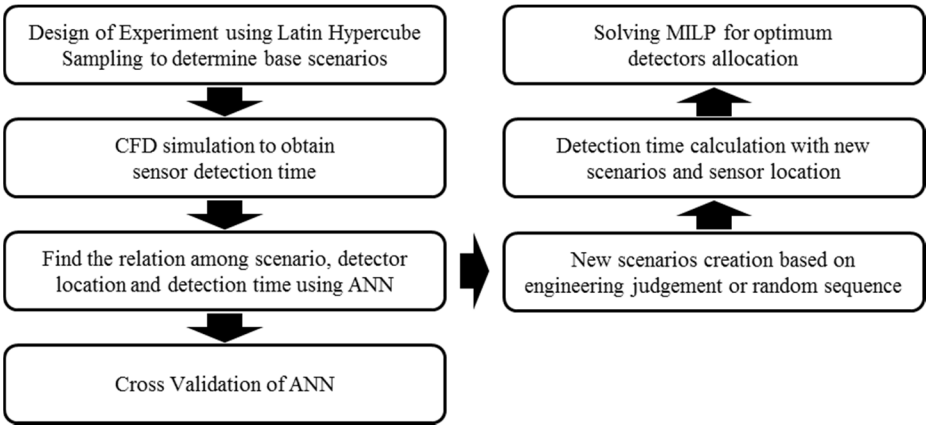

Figure 17 Process of detector layout optimization with metamodeling

4.3.1. CFD Setting

As scenario inputs, there are total five variables that can be classified into two categories. U_x and U_y are meteorological inputs, which are wind velocity in x and y direction respectively. The input value of wind in each direction is from min -3.5m/s to max 3.5m/s and can increase up to 5m/s in the simulation. The leak variables are $\dot{m}, S_x, S_y$ and each represents the mass rate (kg/s), the x-coordinate (m) of the leak point, and the y-coordinate (m) of the leak point.

The results of the Latin Hypercube Sampling (LHS) can be shown in Figure 18. Each point means the leak source location and the size of the point means the amount of leak rate of LNG. The length and direction of the arrow in the point mean the size and direction of the wind, respectively. In Scenarios 6 and 27, the y coordinates are moved to -4 m and +4 m, because of the overlap of the leak point and the obstacle in the geometry. Details of the data sample are listed in Table 5.

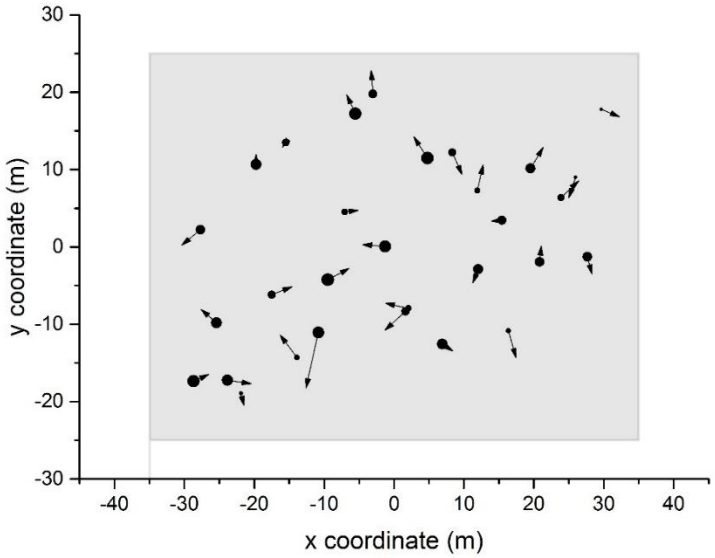

Figure 18 LHS Result for 5 variables

The geometry for the LNG storage can be shown in Figure 19, which is derived from the pool spread simulation included in FLACS. The methane, ethane and propane volume composition of the LNG used in the simulation is 95:4:1. The grid used in the simulation is set to 1m, 1m, and 1m in the x, y, and z directions in the core region and grows by a factor of 1.2 which generate 196,236 cells. The potential detector has a total of 1,587 positions ((23, 23, 3) in the (x, y, z) direction). Some of the 1,587 detectors succeed in detecting the leak, and some fail. Scenarios that are detected successfully are defined as detected scenarios, and scenarios that are not detected are defined as undetected scenarios.

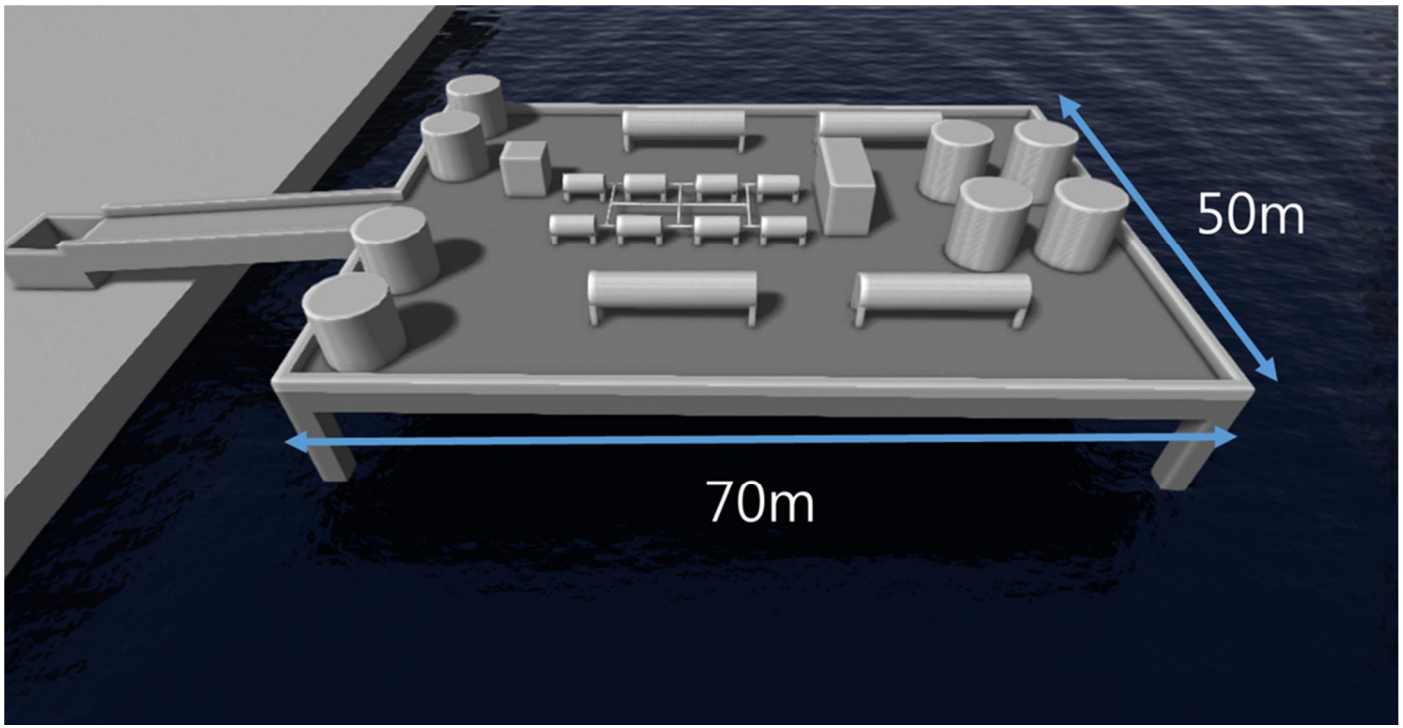

Figure 19 Geometry of an LNG storage

4.3.2. ANN Regression

The process of training the ANN using the simulation results is done through MATLAB. One of the important factors for regression is the number of hidden layers. Too many hidden layers increase the likelihood of overfitting the model, while too few hidden layers do not catch major variables. The number of hidden layers used in this paper is set to 10. Because the scenarios to be generated through the final surrogate model are detected scenarios, the data set used for regression is only detected scenarios. Undetected scenarios are excluded because they unnecessarily increase the amount of data and reduce the quality of the regression. Instead, it is used later for cross-validation. Eight variables including the x, y, and z coordinates of the sensor position and five input variables of the simulation are used as the input vector of the ANN. The structure of the ANN is shown in Figure 20.

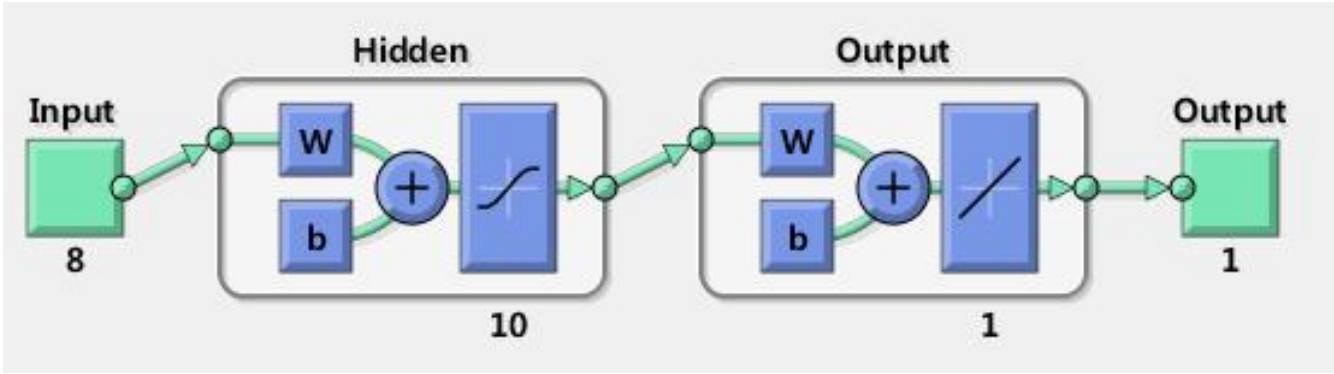

Figure 20 Structure of the ANN

First, train ANNs with all detected scenarios to see if patterns can be caught. Then test the accuracy with 30 base scenarios using Leave-One-Out Cross-Validation (LOOCV) which is useful method when the number of data sets is small. Finally, all the data sets of 47610 (30 base scenarios × 1587 potential sensor position) including detected and undetected are used to classify the undetected scenarios to see if they affect the quality of the ANN. And then, extended scenarios are created through the model with newly generated inputs. One of the advantages of the surrogate model is that it takes only a few minutes to generate extended scenarios via this ANN due to the low computational cost.

4.3.3. Detector Allocation Optimization

In this chapter, Detector allocation optimization is solved with MILP. The equations for MILP follows S.W. Legg (2012) and details are described below. The variables are summarized in table 4.

$$\min \sum_{a\in A} \alpha_a \sum_{i\in \mathcal{L}_a} d_{a,i} x_{a,i} \tag{6}$$

s.t.

$$\sum_{l\in \mathcal{L}} s_l \leq p \tag{7}$$

$$x_{a,i} \leq s_l \qquad \forall\, a \in A,\ i \in \mathcal{L}_a \tag{8}$$

$$\sum_{i\in \mathcal{L}_a} x_{a,i} = 1 \qquad \forall\, a \in A \tag{9}$$

$$s_l \in \{0,1\} \qquad \forall\, l \in L \tag{10}$$

$$0 \leq x_{a,i} \leq 1 \qquad \forall\, a \in A,\ i \in \mathcal{L}_a \tag{11}$$

The objective function for finding the minimum value of the mean detection time for each scenario a is shown in Equation 6. In the MILP, it is assumed that all scenarios happen with equal probability, which is $\alpha_a = 1/M$. The term $d_{a,i}$ is the expected damage coefficient, which can be defined as the expected damage in the explosion scenario and the expected fatality in the toxic gas scenario. Here, $d_{a,i}$ is defined as the time sensor i senses gas in scenario a, and pre-calculated values are input in matrix form. The term s_l is a binary variable and has a value of 1 if the sensor exists in the potential position l or 0 otherwise. In scenario a, the sensor at position i where gas is first detected can be represented by $x_{a,i}$. It is expressed as an integer by

Eqs. (8), (9) and (10) and has a value of 0 or 1. The maximum number of sensors is limited by Eqs. (7). In equation (9), the number of sensors that detect gas for the first time is set to 1

Table 4 Problem notation

SYMBOL	MEANING
$\mathbf{A} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \dots, \mathbf{M}\}$	Leak scenarios
$\mathbf{L} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \dots, \mathbf{N}\}$	Potential detector locations
$\boldsymbol{\mathcal{L}_a}$	Sensor locations affected by scenario a
$\boldsymbol{\alpha_a}$	The probability of leak scenario a
$\boldsymbol{d_{a,i}}$	Damage coefficient for leak scenario a at location i
$\boldsymbol{s_l}$	Binary variable indicating whether a sensor is installed at location l or not
$\mathbf{p}$	Maximum number of detectors
$\boldsymbol{x_{a,i}}$	Indicator for location i that first detects scenario a

4.4. Result

4.4.1. ANN results

A total of 47610 datasets are obtained after running all the cases (30 scenarios × 1587 potential sensor locations). All detection times obtained are converted to log-scale. Figure 21 means the error histogram between the predicted detection time and the actual detection time, showing that ANNs are well trained without skewness. As a result of the comparison, most of the data are distributed between 0.2 deviations and the error was 1.6 times smaller compared to the normal scale.

Table 5 below shows the LOOCV result Mean Square Error (MSE) values for each scenario with detailed information for all scenarios. The overall MSE for the model is 0.0419, and the predicted values of each scenario based on the rest of the scenarios show a high performance ranging between 0.0411 and 0.0652.

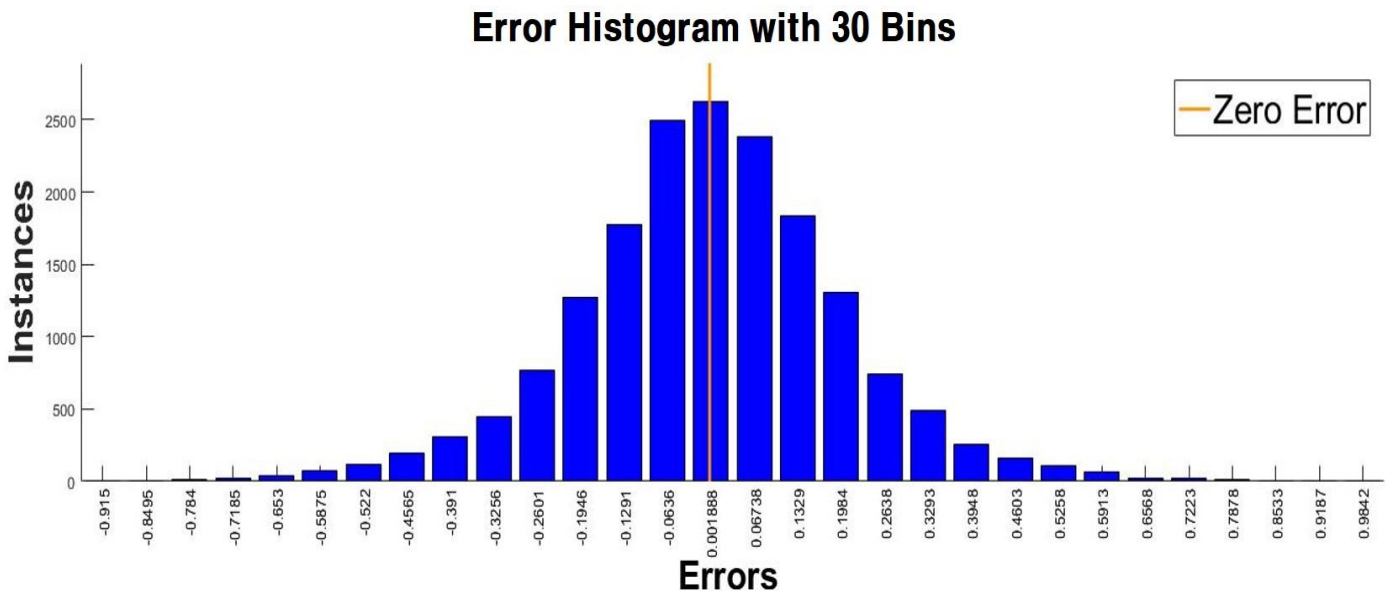

Figure 21 Error histogram of the trained network

Table 5 Cross validation result

Scenario	*Mass*	*Sx*	*Sy*	*Ux*	*Uy*	*MSE*
Unit	kg/s	m	m	m/s	m/s	
1	11.54	-21.90	-18.94	0.43	-1.56	0.0511
2	10.23	25.98	8.99	-0.98	-2.64	0.0495
3	55.00	27.66	-1.26	0.66	-2.23	0.0450
4	40.85	-17.51	-6.19	2.92	1.02	0.0513
5	38.71	8.33	12.23	1.40	-2.89	0.0453
6	95.45	4.79	11.48	-1.98	-1.22	0.0542
7	46.46	-3.04	19.78	-0.24	3.05	0.0460
8	18.19	-13.89	-14.32	-2.44	2.93	0.0469
9	35.43	-15.48	13.50	-0.50	-0.59	0.0420
10	75.51	-25.40	-9.81	-2.23	1.72	0.0460
11	99.08	-9.49	-4.24	3.08	1.48	0.0446
12	73.74	-19.75	10.67	-0.02	1.30	0.0457
13	26.23	2.07	-7.95	-3.32	0.66	0.0453
14	65.09	12.04	-2.87	-0.76	-1.80	0.0489
15	78.09	-23.84	-17.24	3.42	-0.45	0.0587
16	50.33	15.43	3.43	-1.50	-0.09	0.0460
17	32.76	23.88	6.37	2.54	2.18	0.0483
18	16.09	16.36	-10.85	1.07	-3.47	0.0411
19	43.66	1.64	-8.35	-2.96	-2.45	0.0411
20	22.80	11.93	7.27	0.81	3.35	0.0556
21	89.81	-5.54	17.22	-1.25	2.46	0.0407
22	28.04	-7.08	4.54	1.95	0.21	0.0425
23	91.22	-28.66	-17.37	2.17	0.91	0.0652
24	56.64	20.84	-1.94	0.20	2.07	0.0453
25	60.04	-27.71	2.22	-2.67	-1.97	0.0443
26	84.92	-1.28	0.08	-3.25	0.25	0.0440
27	82.55	-10.84	-11.07	-1.77	-3.18	0.0447
28	70.46	6.91	-12.57	1.50	-0.92	0.0411
29	8.16	29.64	17.79	2.62	-0.95	0.0434
30	66.49	19.53	10.16	1.81	2.73	0.0445

4.4.2. Sensor Allocation Result

The MILP determines the binary parameter, s_l, indicating the presence or absence of sensor at the potential location and the total number of variables to be determined is 1587. The number of coefficients to be covered is massive especially in the inequality constraints. A sparse coefficient matrix is used in MATLAB because it is difficult to directly handle this coefficient matrix.

The detector allocation optimization results are shown in Figure 22. The figure includes 30 basic scenarios and 130 extended scenarios. With 30 scenarios, as the number of detectors increases, the minimum detection time converges to 5 seconds, the start time of the leak, and immediately detects a leak. Using 130 extended scenarios shows similar tendency but the detection time is bigger than the leak start point because it has to cover more scenarios than the number of detectors.

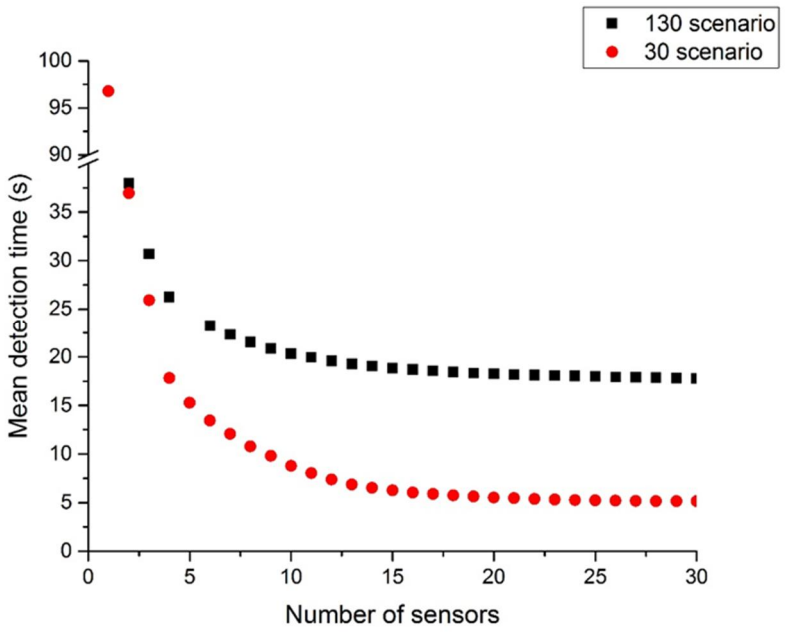

Figure 22 Detection time by the number of sensors

4.4.3. Discussion

Two subjects should be discussed to show this surrogate model based optimization is practical. First, it is necessary to evaluate the model, including data sets for undetected scenario because they are excluded in the regression. Although there is some distribution between the detection time in the sample and the model, the percentage of misclassification is low whether the gas is detected within a certain time or not. A confusion matrix is used to evaluate the performance of the model to classify detected and undetected scenarios. And 30 seconds is used as a criterion for classification. Similar to the regression problem, the number of hidden layers is 10. Performance of the training is calculated with the cross-entropy term using the scaled conjugate gradient training method. 70% of the datasets are used in the training set, 15% in the validation set, and 15% in the test set. The results are shown in Table 6. Most of the scenarios are classified as detected and undetected within the 30 seconds, which means ANN training is well-defined even after including the undetected scenarios.

Table 6 Confusion matrix of the trained network

		Output Class		Overall Accuracy
		Not Detect (0)	Detect (1)	
Target Class	Not Detect (0)	True Negative 44141 (92.7%)	False Positive 19 (0.0%)	100.0%
	Detect (1)	False Negative 27 (0.1%)	True Positive 3423 (7.2%)	99.2%
Overall Accuracy		99.9%	99.4%	99.9%

The second problem is whether the generated extended scenarios can be applied to the entire process and thus the optimization can be changed. Figure 23 and 24 show that the detectors are away from the source location with 130 scenarios unlikely the 30 scenarios case.

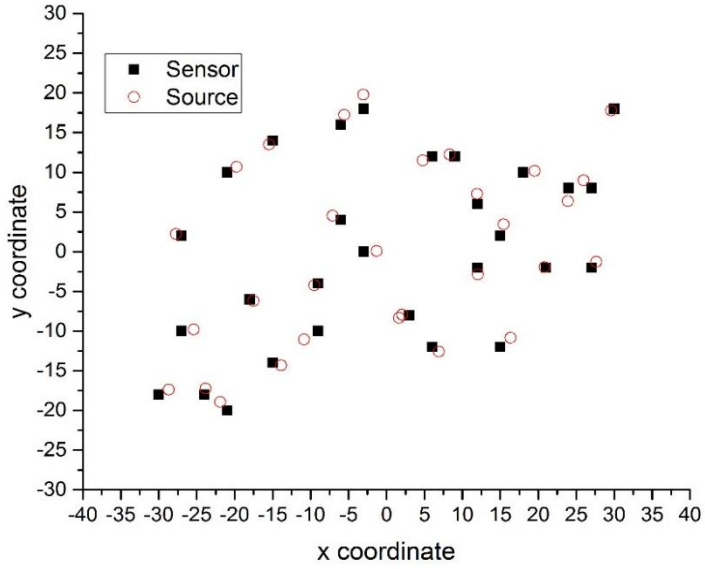

Figure 23 Optimization result with 30 scenarios

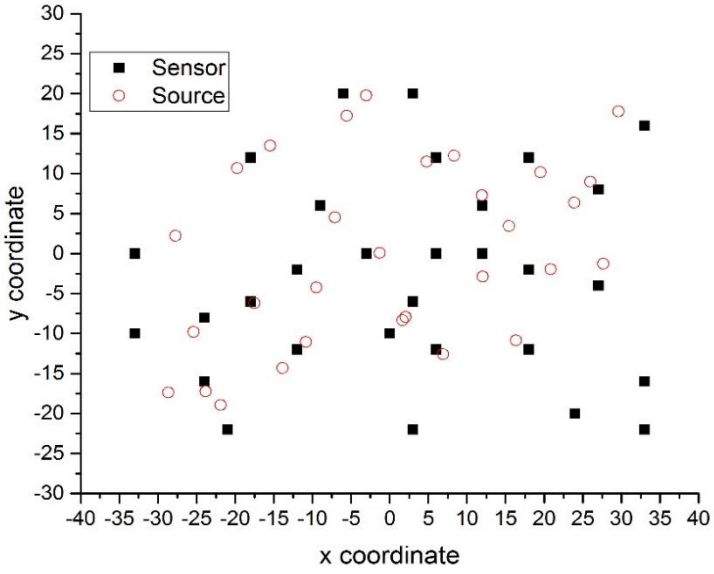

Figure 24 Optimization result with 130 scenarios (red circles are the 30 source locations in the base scenarios)

4.5. Chapter Conclusion

In this chapter, the detector allocation optimization with the surrogate model is dealt. With the surrogate model, the number of scenarios needed for sensor layout optimization greatly increases, which increased the reliability of the optimization. Also, by using this model, we can know the point where the efficiency of the number of sensors becomes low and the point for mean detection time to converge. It can help Computational Fluid Dynamics to become a useful tool in the chemical safety field that needs a lot of computational costs. QRA (Quantitative Risk Analysis) and real-time dispersion prediction combined with model reduction technics would be promising future. And the development of sensor allocation optimization techniques is applicable in the future.

Chapter 5: Concluding remarks

5.1 Conclusion

This thesis has addressed the hazardous gas dispersion surrogate model based on Computational Fluid Dynamics and it`s applications. Toxic gas dispersion model, whole process to develop gas dispersion surrogate model and gas sensor layout optimization have been considered.

At first, TOXIM, a specialized simulator for hazardous gas dispersion and risk analysis, was developed. The robustness of the calculation was confirmed by conducting a case study according to the change of wind direction, wind speed, and species. In addition, the accuracy of the model was verified compared to the actual leak accident.

Secondly, the CFD model was reduced and regressed through a surrogate model based on variational autoencoder with deep convolutional layer interconnected with a deep neural network (VAEDC-DNN). The mse of suggested model was 0.00246. The calculation time required to obtain the final result was reduced from several hours to less than a second. Also, it is confirmed that image generation is not overfitting by data memorization through the smoothness of image transition in the variable space.

Lastly, the position of the gas sensor was optimized through a deep neural network based surrogate model and mixed integer linear programming. A more reliable detector location was found by amplifying the data sample using the surrogate model.

5.1 Future works

For the future, various discharge models will be added to the currently developed simulator TOXIM. In addition to the current gas discharge model, two phase discharge, pool spread and evaporation models can be applied.

Also, it can be expected a better performance if the surrogate model is developed using the state-of-the-art model. For example, if generative models are used such as boundary equilibrium generative adversarial networks (BEGAN), more accurate images can be generated even in more complex structures.

Finally, the currently developed image generating model is an instantaneous model that can only obtain results at a fixed time, but in the future, a time continuous model will be developed to identify the dispersion path of hazardous gas over time.

Nomenclature

NOMENCLATURE

P_{death}	Probability of death
β_v	volume porosity
ρ	density(kg/m^3)
β_j	area porosity in the *j* th direction (m/s)
u_j	mean velocity (*j* th component, vector) (m/s)
$\dot{m}$	mass rate or release rate (kg/s)
V	volume (m^3)
P	gauge pressure, overpressure (Pa)
σ_{ij}	stress tensor (N/ m^2)
$F_{o,i}$	flow resistance created by sub-grid obstacles (N)
$F_{w,i}$	flow resistance created by walls (N)
ρ_0	density of sub grid object(kg/m^3)
g_i	gravitational acceleration in the *i* th direction (m/s^2)
δ_{ij}	Kronecker delta function
μ_{eff}	effective viscosity (Pa·s)
μ	turbulent viscosity (Pa·s)
C_μ	constant in the k-ε equation; typically C_μ=0.09
ε	dissipation of turbulent kinetic energy (m^2/s^3)
k	turbulent kinetic energy (m^2/s^2)
Pr	probit
c	concentration in ppm by volume
x	P_{death} contour data
z	latent space
N_z	the number of latent variables
$p_\theta(x\|z)$	probabilistic decoder as generator with parameter θ
$q_\phi(z\|x)$	probabilistic encoder with parameter ϕ
v	variable space
$f^*(v)$	surrogate model function

$f(v)$	CFD model function
$\mathcal{L}(\theta, \phi; x^{(i)})$	variational lower bound
D_{KL}	Kullback-Leibler divergence
$l(\theta, \phi; x^{(i)})$	loss function
N_{train}	the number of training data set
N_v	the number of variables
L	Potential detector locations
A	Leak scenarios
$\mathcal{L}_a$	Sensor locations affected by scenario a
α_a	The probability of leak scenario a
$d_{a,i}$	Damage coefficient for leak scenario a at location i
p	Maximum number of detectors
s_l	Binary variable indicating whether a sensor is installed at location l or not
$x_{a,i}$	Indicator for location i that first detects scenario a

Abbreviations

CFD	computational fluid dynamics
VAEDC	variational autoencoder with deep convolutional layers
DNN	deep neural network
PCA	principal component analysis
segPCT	segmented principal component transform-principal component analysis
CNN	convolutional neural network
BN	batch normalization
Adam	adaptive momentum estimation
RANS	Reynolds-averaged Naiver-Stokes
AEGLs	Acute Exposure Guideline Levels
ERPGs	Emergency Response Planning Guidelines
TEELs	Temporary Emergency Exposure Limits
CFL	Courant-Friedrich-Levy
NN	Neural network

AE	autoencoder
DAE	deep autoencder
DCAE	deep convolutional autoencdoer
VAE	variational autoencoder
ANN	Artificial Neural Network
DoE	Design of Experiment
FLACS	Flame Acceleration Simulator
LFL	Lower Flammability Limit
LHS	Latin Hypercube Sampling
LNG	Liquefied Natural Gas
LOOCV	Leave one out cross-validation
MILP	Mixed Integer Linear Programming
MSE	Mean Square Error
QRA	Quantitative Risk Analysis
RANS	Reynolds Averaged Navier Stokes

Subscripts and superscripts

train	training set
validation	validation set
test	test set
gen	generated data

Literature cited

▶ Chapter 1

Hjertager, B. (1986). Three-dimensional modeling of flow, heat transfer, and combustion. Handbook of heat and mass transfer, 1, 1303-1350.

Launder, B. E., & Spalding, D. B. (1974). The numerical computation of turbulent flows. Computer Methods in Applied Mechanics and Engineering, 3, 269-289.

Na, J., Jeon, K., Lee, W. B. (2018). Toxic Gas Release Modeling for Real-time Analysis Using Variational Autoencoder with Convolutional Neural Networks. Chemical Engineering Science, 181, 68-78

Yang, S., Jeon, K., Kang, D., Han, C. (2017). Accident Analysis of the Gumi Hydrogen Fluoride Gas Leak Using CFD and Comparison with Post-Accidental Environmental Impacts. Journal of Loss Prevention in the Process Industries. 48, 207-215

▶ Chapter 2

Gu, S., I. Choi, W. Kim, O. Sun, S. Kim and Y. Lee (2013). "Study on the Distribution of Fluorides in Plants and the Estimation of Ambient Concentration of Hydrogen Fluoride Around the Area of the Accidental Release of Hydrogen Fluoride in Gumi." Korean Journal of Environmental Health Sciences 39(4): 346-353.

Joo, H. S. (2013). A study on the improvement of environmental impact assessment of industrial complexes based on risk assessment of

chemical leakeage accidents. K. E. Institute.

Koh, D. H. (2014). A study on the establishment of environmental impact area of hydrofluoric acid spill in Gumi, Korea, Seoul National University.

Korea Occupational Safety & Health Agency (2013). Case study of hydrogen fluoride accidental release (in Korean).

Lee, M.-R., S. Koo and J.-H. Shim (2013). "Assessment of Estimated Damage Area by CCTV Images: Case Study of Gumi Hydrofluoric Acid Gas Leakage." Journal of Korean Society of Hazard Mitigation 13(6): 223-229.

Yim, B. and S.-T. Kim (2016). "Estimation of the Concentration of HF in the Atmosphere Using Plant Leaves Exposed to HF in the Site of the HF Spill." Journal of Korean Society for Atmospheric Environment 32(3): 248-255. DOI : 10.5572/KOSAE.2016.32.3.248

▶ Chapter 3

Barratt, R. (2013). Atmospheric dispersion modelling: an introduction to practical applications: Routledge.

Broughton, E. (2005). The Bhopal disaster and its aftermath: a review. Environmental Health, 4, 6.

Caballero, J. A., & Grossmann, I. E. (2008a). An algorithm for the use of surrogate models in modular flowsheet optimization. AIChE Journal, 54, 2633-2650.

Caballero, J. A., & Grossmann, I. E. (2008b). Rigorous flowsheet optimization using process simulators and surrogate models. Computer Aided

Chemical Engineering, 25, 551-556.

Chen, T., Hadinoto, K., Yan, W., & Ma, Y. (2011). Efficient meta-modelling of complex process simulations with time–space-dependent outputs. Computers & Chemical Engineering, 35, 502-509.

Delaunay, D. (1996). Numerical simulation of atmospheric dispersion in an urban site: Comparison with field data. Journal of Wind Engineering and Industrial Aerodynamics, 64, 221-231.

Dharmavaram, S., Hanna, S. R., & Hansen, O. R. (2005). Consequence analysis—Using a CFD model for industrial sites. Process Safety Progress, 24, 316-272.

Géron, A. (2017). Hands on Machine Learning with scikit-learn and Tensorflow. In: O'Reilly Media.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In T. Yee Whye & T. Mike (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Vol. 9, pp. 249--256). Proceedings of Machine Learning Research: PMLR.

Gomes, M. V. C., Bogle, I. D. L., Biscaia, E. C., & Odloak, D. (2008). Using kriging models for real-time process optimisation. Computer Aided Chemical Engineering, 25, 361-366.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. In: MIT Press.

Ha, D., Park, D., Koo, J., Baek, K. H., & Han, C. (2016). Improvement of principal component analysis modeling for plasma etch

processes through discrete wavelet transform and automatic variable selection. Computers & Chemical Engineering, 94, 362-369.

Hanna, S., & Strimaitis, D. (1988). Workbook of test cases for vapor cloud source emission and dispersion models. CCPS/AIChE, New York, NY.

Hanna, S. R., Brown, M. J., Camelli, F. E., Chan, S. T., Coirier, W. J., Kim, S., Hansen, O. R., Huber, A. H., & Reynolds, R. M. (2006). Detailed Simulations of Atmospheric Flow and Dispersion in Downtown Manhattan: An Application of Five Computational Fluid Dynamics Models. Bulletin of the American Meteorological Society, 87, 1713-1726.

Hanna, S. R., Hansen, O. R., & Dharmavaram, S. (2004). FLACS CFD air quality model performance evaluation with Kit Fox, MUST, Prairie Grass, and EMU observations. Atmospheric Environment, 38, 4675-4687.

Hanna, S. R., Hansen, O. R., Ichard, M., & Strimaitis, D. (2009). CFD model simulation of dispersion from chlorine railcar releases in industrial and urban areas. Atmospheric Environment, 43, 262-270.

Hansen, O. R., Gavelli, F., Ichard, M., & Davis, S. G. (2010). Validation of FLACS against experimental data sets from the model evaluation database for LNG vapor dispersion. Journal of Loss Prevention in the Process Industries, 23, 857-877.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In ArXiv e-prints (Vol. 1502).

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. Science, 313, 504-507.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In ArXiv e-prints (Vol. 1502).

Jin, H. D., Lee, Y.-H., Lee, G., & Han, C. (2006). Robust Recursive Principal Component Analysis Modeling for Adaptive Monitoring. Industrial & Engineering Chemistry Research, 45, 696-703.

Kajero, O. T., Chen, T., Yao, Y., Chuang, Y.-C., & Wong, D. S. H. (2017). Meta-modelling in chemical process system engineering. Journal of the Taiwan Institute of Chemical Engineers, 73, 135-145.

Kano, M., Hasebe, S., Hashimoto, I., & Ohno, H. (2001). A new multivariate statistical process monitoring method using principal component analysis. Computers & Chemical Engineering, 25, 1103-1113.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. In ArXiv e-prints (Vol. 1412).

Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. In ArXiv e-prints (Vol. 1312).

Kresta, J. V., Macgregor, J. F., & Marlin, T. E. (1991). Multivariate statistical monitoring of process operating performance. The Canadian Journal of Chemical Engineering, 69, 35-47.

Lee, J.-M., Qin, S. J., & Lee, I.-B. (2006). Fault detection and diagnosis based on modified independent component analysis. AIChE Journal, 52, 3501-3514.

Long, K. J., Zajaczkowski, F. J., Haupt, S. E., & Peltier, L. J. (2009). Modeling a Hypothetical Chlorine Release on a College Campus. JCP, 4, 881-

890.

Loy, Y. Y., Rangaiah, G. P., & Lakshminarayanan, S. (2017). Surrogate modelling for enhancing consequence analysis based on computational fluid dynamics. Journal of Loss Prevention in the Process Industries, 48, 173-185.

Masci, J., Meier, U., Cireşan, D., & Schmidhuber, J. (2011). Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. In T. Honkela, W. Duch, M. Girolami & S. Kaski (Eds.), Artificial Neural Networks and Machine Learning – ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I (pp. 52-59). Berlin, Heidelberg: Springer Berlin Heidelberg.

Middha, P., Hansen, O. R., Grune, J., & Kotchourko, A. (2010). CFD calculations of gas leak dispersion and subsequent gas explosions: Validation against ignited impinging hydrogen jet experiments. Journal of Hazardous Materials, 179, 84-94.

Misra, M., Yue, H. H., Qin, S. J., & Ling, C. (2002). Multivariate process monitoring and fault diagnosis by multi-scale PCA. Computers & Chemical Engineering, 26, 1281-1293.

Palmer, K., & Realff, M. (2002). Optimization and Validation of Steady-State Flowsheet Simulation Metamodels. Chemical Engineering Research and Design, 80, 773-782.

Perry, W. W., & Articola, W. P. (1980). Study to modify the vulnerability model of the risk management system. In: ENVIRO CONTROL INC ROCKVILLE MD.

Qin, S. J. (2012). Survey on data-driven industrial process monitoring and diagnosis. Annual Reviews in Control, 36, 220-234.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In ArXiv e-prints (Vol. 1511).

Wang, K., Chen, T., Kwa, S. T., Ma, Y., & Lau, R. (2014). Meta-modelling for fast analysis of CFD-simulated vapour cloud dispersion processes. Computers & Chemical Engineering, 69, 89-97.

Wilson, Z. T., & Sahinidis, N. V. (2017). The ALAMO approach to machine learning. Computers & Chemical Engineering, 106, 785-795.

Withers, R., & Lees, F. (1985a). The assessment of major hazards: The lethal toxicity of chlorine: Part 1, Review of information on toxicity. Journal of Hazardous Materials, 12, 231-282.

Withers, R., & Lees, F. (1985b). The assessment of major hazards: The lethal toxicity of chlorine: Part 2, Model of toxicity to man. Journal of Hazardous Materials, 12, 283-302.

Xie, Z.-T., Hayden, P., & Wood, C. R. (2013). Large-eddy simulation of approaching-flow stratification on dispersion over arrays of buildings. Atmospheric Environment, 71, 64-74.

Yang, S., Jeon, K., Kang, D., & Han, C. (2017). Accident analysis of the Gumi hydrogen fluoride gas leak using CFD and comparison with post-accidental environmental impacts. Journal of Loss Prevention in the Process Industries, 48, 207-215.

▶ Chapter 4

Benavides-Serrano, A., Legg, S., Vázquez-Román, R., Mannan, M., & Laird, C. (2013). A stochastic programming approach for the optimal placement of gas detectors: unavailability and voting strategies. Industrial & Engineering Chemistry Research, 53, 5355-5365.

Benavides-Serrano, A., Mannan, M., & Laird, C. (2015). A quantitative assessment on the placement practices of gas detectors in the process industries. Journal of Loss Prevention in the Process Industries, 35, 339-351.

Benavides-Serrano, A., Mannan, M., & Laird, C. (2016). Optimal placement of gas detectors: AP-median formulation considering dynamic nonuniform unavailabilities. AIChE Journal, 62, 2728-2739.

Berry, J., Hart, W. E., Phillips, C. A., Uber, J. G., & Watson, J.-P. (2006). Sensor placement in municipal water networks with temporal integer programming models. Journal of water resources planning and management, 132, 218-224.

Davis, S., Hansen, O. R., Gavelli, F., & Bratteteig, A. (2015). Using CFD to Analyze Gas Detector Placement in Process Facilities. In: GexCon.

Gomes, E. G., de Andrade Medronho, R., & Alves, J. V. B. (2014). Gas Detector Placement in Petroleum Process Unit Using Computational Fluid Dynamics. International Journal of Modeling and Simulation for the Petroleum Industry, 8.

Hamel, D., Chwastek, M., Farouk, B., Dandekar, K., & Kam, M. (2006). A computational fluid dynamics approach for optimization of a sensor

network. In Measurement Systems for Homeland Security, Contraband Detection and Personal Safety, Proceedings of the 2006 IEEE International Workshop on (pp. 38-42): IEEE.

Hanna, S. R., Brown, M. J., Camelli, F. E., Chan, S. T., Coirier, W. J., Kim, S., Hansen, O. R., Huber, A. H., & Reynolds, R. M. (2006). Detailed Simulations of Atmospheric Flow and Dispersion in Downtown Manhattan: An Application of Five Computational Fluid Dynamics Models. Bulletin of the American Meteorological Society, 87, 1713-1726.

Hanna, S. R., Hansen, O. R., Ichard, M., & Strimaitis, D. (2009). CFD model simulation of dispersion from chlorine railcar releases in industrial and urban areas. Atmospheric Environment, 43, 262-270.

Legg, S., Benavides-Serrano, A., Siirola, J. D., Watson, J.-P., Davis, S., Bratteteig, A., & Laird, C. D. (2012). A stochastic programming approach for gas detector placement using CFD-based dispersion simulations. Computers & Chemical Engineering, 47, 194-201.

Legg, S., Wang, C., Benavides-Serrano, A., & Laird, C. (2013). Optimal gas detector placement under uncertainty considering Conditional-Value-at-Risk. Journal of Loss Prevention in the Process Industries, 26, 410-417.

Long, K. J., Zajaczkowski, F. J., Haupt, S. E., & Peltier, L. J. (2009). Modeling a Hypothetical Chlorine Release on a College Campus. JCP, 4, 881-890.

Loy, Y., Rangaiah, G., & Lakshminarayanan, S. (2017). Surrogate modelling for enhancing consequence analysis based on computational fluid

dynamics. Journal of Loss Prevention in the Process Industries, 48, 173-185.

Margheri, L., & Sagaut, P. (2016). A hybrid anchored-ANOVA–POD/Kriging method for uncertainty quantification in unsteady high-fidelity CFD simulations. Journal of Computational Physics, 324, 137-173.

Vázquez-Román, R., Díaz-Ovalle, C., Quiroz-Pérez, E., & Mannan, M. S. (2016). A CFD-based approach for gas detectors allocation. Journal of Loss Prevention in the Process Industries, 44, 633-641.

Wang, K., Chen, T., Kwa, S. T., Ma, Y., & Lau, R. (2014). Meta-modelling for fast analysis of CFD-simulated vapour cloud dispersion processes. Computers & Chemical Engineering, 69, 89-97.

Xie, Z.-T., Hayden, P., & Wood, C. R. (2013). Large-eddy simulation of approaching-flow stratification on dispersion over arrays of buildings. Atmospheric Environment, 71, 64-74.

Abstract in Korean (요 약)

기계 학습을 활용한 유해 가스 확산 대리 모델의 개발과 응용

전산유체역학(CFD)을 이용하여 유해 화학 가스의 확산을 예측하는 방법은 높은 정확도로 인하여 다양한 방법으로 활용할 수 있다는 장점이 있다. 하지만 구조적인 복잡성과 많은 계산 자원을 필요로 한다는 점이 그 활용에 걸림돌이 된다.

먼저 전산유체역학의 구조적인 복잡성을 해결하여 누구나 쉽게 사용할 수 있는 가스 확산 시뮬레이터인 TOXIM을 개발하였다. TOXIM을 활용하여 다양한 조건에서 케이스 스터디를 수행하여 시뮬레이터의 계산 안정성을 확인하였다. 또한 구미에서 실제로 발생한 누출사고를 TOXIM의 방법론으로 모사한 결과 실제와 모델이 유사한 결과를 나타내어 그 정확도를 입증하였다.

다음으로 전산유체역학의 계산 시간을 줄여 실시간으로 이를 활용하기 위하여 대리 모델을 개발하였다. 다양한 딥러닝 기법을 사용하여 전산유체역학 계산 결과를 압축한 뒤 신경망 기반의 함수를 사용하여 대리모델을 만들었다. 그 중 VAEDC-DNN기법이 가장 높은 정확도를 보였다. 계산 시간 또한 실제 전산유체역학은 수시간이 걸리는 것에 비해 대리 모델을 사용하였을 때 초단위로 줄어들

었음을 확인하였다. 이를 통하여 전산유체역학을 가스 누출 사고 발생시 실시간으로 활용할 수 있음을 입증하였다.

마지막으로 전산유체역학 대리 모델을 가스 누출 감지기 배치 문제에 적용하였다. 대리 모델을 사용하여 여러 가지 가스 누출 사고 시나리오를 생성하였고, 이렇게 생성된 샘플을 활용하여 정수혼합 선형최적화를 적용하여 감지기의 위치를 최적화하였다. 대리 모델을 활용하여 최적화를 진행한 경우와 그렇지 않은 경우를 비교한 결과 대리모델을 활용한 쪽이 더 좋은 결과를 보임을 확인하였다.

주요어: 전산유체역학, 대리 모델, 유해 가스 확산, 기계 학습, 딥러닝

학번: 2014-21541

성명: 전 경 우