FVM Network to Reduce Computational Cost of CFD Simulation

Joongoo Jeon¹, Sung Joong Kim²

¹jgjeon@hanyang.ac.kr, ²sungjkim@hanyang.ac.kr *Hanyang University*

Abstract - Despite the rapid growth of CPU performance, the computational cost to simulate the chemically reacting flow is still infeasible in many cases. There are few studies to accelerate the CFD simulation by using neural network models. However, they noted that it is still difficult to predict multi-step CFD time series data. The finite volume method (FVM) which is the basic principle of most CFD codes seems not to be sufficiently considered in the previous network models. In this study, a FVM network (FVMN) which simulate the principles of FVM by the tier-input and derivativeoutput system was proposed. The performance of this baseline model was evaluated using unsteady reacting flow datasets. It was confirmed that the maximum relative error of the FVMN (0.04%) was much smaller than the general model (1.12%) in the training dataset. This difference in error size was more prominent in the prediction datasets. In addition, it was observed that the calculation speed was about 10 times faster in FVMN than CFD solver even under the same CPU condition. Although the relative error with the ground truth data was significantly reduced in the proposed model, the linearly increasing gradient error is a remaining issue in longer transient calculations. Therefore, we additionally suggested Machine learning aided CFD framework which can substantially accelerate the CFD simulation through alternating computations.

Keywords- Machine learning, deep learning, baseline model, CFD, finite volume method.

1. INTRODUCTION

In recent years, many computational studies to investigate the reacting flows of flammable gases have been conducted in various research fields [1-3]. Especially, the potential of hydrogen explosion is being vigorously simulated by various CFD codes as hydrogen energy has emerged as a promising alternative to replace traditional carbon-based energy sources [4-6]. The released hydrogen gas from the hydrogen facility can be ignited depending on mixing conditions [7]. If the ignited

gas is not extinguished prematurely, it can lead to an explosion in flame acceleration friendly geometry conditions. The prediction of this series of flame propagation process requires sophisticated CFD simulation [5].

Despite the rapid growth of CPU performance, the computational cost to simulate the chemically reacting flow is still infeasible in many cases [5]. This is especially true for turbulent reacting flow accompanied by less than mm scale for grid and ms scale for timestep [5]. These physical constraints can be confirmed by a recent study of Tolias et al.'s reacting flow simulations [4]. They investigated the ability of current CFD codes to predict hydrogen deflagrations in vented enclosures. They identified that about 10 - 100 h CPU time per 1 s physical time was required to simulate unsteady hydrogen deflagration through CFD simulations, based on a domain size of $20.0 \times 14.4 \times 12.0 m$ [4]. In nuclear industry, these costly simulations are impractical given that 72 h of accident analysis is stated in the regulatory guide [8].

As a solution to reduce the computational cost, we paid attention to machine learning technique which is in the limelight as the most innovative technology. Advanced machine learning technique is also being actively applied in physics, especially to calculate constitutive and governing equations. Chen et al. developed a network model to solve ODE for time series prediction, with flexibility in the modeling of irregularity [9]. Kim et al. proposed a scale separation method to training Neural ODE for stiff system corresponding to reacting flow problem [10]. Ledesma et al. suggested a neural network topology to estimate the derivative of a performance function [11]. However. the evaluation of these models is still limited to a single differential equation. There are still few studies to predict the time series data of CFD simulation involving multiple differential equations. Takbiri et al. investigated predictive capability of machine learning technique for DNS simulations [12]. Lee et al. compared the performance of each network model with unsteady flow simulations [13].

However, they noted that it was still difficult to predict multi-step flow field [13]. As the time distance between the training point and prediction point increases, the error increases rapidly.

The aforementioned literatures highlight that the improvement of the network models to predict the CFD time series data more accurately is still required. Especially, the finite volume method (FVM) which is the basic principle of most CFD codes seems not to be sufficiently considered in the previous models. Most recently, Praditia et al. suggested a network model adopting the numerical structure of the FVM [14]. This model of solving PDE by constructing a network model for each term of the governing equation seems intuitive, but this complexity can be disadvantageous in terms of differentiation from the CFD cost, and application in implicit scheme. To improve both computational speed and accuracy of network models, firstly a baseline model which can compare them is required. As many studies emphasized, the establishment of baseline model which performs reasonably well on wide dataset is an essential task [15-17]. Therefore, the objectives of this study were targeted as follows:

 To develop a novel concept of baseline model by thorough understanding of CFD and ML principles.
 To evaluate the performance of the baseline model using on a reacting flow datasets.

Section 2 addresses the motivation and details in the development of a new concept of baseline model. Section 3 introduces the numerical simulation to generate the datasets. Section 4 discusses the results of evaluating the performance of the developed model. Finally, Section 5 summarizes and concludes this paper.

2. Baseline model

2.1. Principles of the FVM

The objective of this study is to develop a novel concept of network model for accurately predicting the CFD time series data. As an application subject, it is important to thoroughly understand the principles of CFD simulation. **Eq.** (1) shows the general transport equation where computational domain is discretized into a finite set of control volumes (grids). ϕ is transported quantity and V is control volume. $\nabla \phi$ is gradient of quantity to calculate the diffusion term.

$$\frac{\partial}{\partial t} \int_{V} \rho \phi \ dV + \int_{V} \nabla \cdot (\rho u \phi) \ dV - \int_{V} \nabla \cdot (\rho \Gamma_{\phi} \nabla \phi) \ dV = \int_{V} S_{\phi}(\phi) \ dV$$

Eq. (2) shows the divergence theorem also called as Gauss's theorem. The theorem demonstrates that the volume integral of the divergence over the region is equal to the surface integral of a vector field over a closed surface (flux) [18].

$$\int_{V} \nabla \cdot a \ dV = \oint_{A} dA \cdot a \tag{2}$$

The basic idea of the FVM, used in most CFD codes, is that divergence terms in the governing equation can be converted to surface integrals using the divergence theorem as shown Eq. (3) [18]. These terms are then calculated as fluxes at each surface. The FVM is a robust conservative method because physical quantity change is calculated by exchanging quantities between neighboring grids [19]. For example, **Eq.** (4) shows the 2D axisymmetric mass continuity equation in the absence of a source term. The physical quantity of the neighboring grid as well as the main grid determines the physical quantity of the main grid in the next timestep [19]. We paid attention to these principles of the FVM, which are interaction with neighboring grid and the update of physical quantity with derivative amount.

$$\frac{\partial}{\partial t} \int_{V} \rho \phi \ dV + \oint_{A} (\rho u \phi) \ dA - \oint_{A} (\rho \Gamma_{\phi} \nabla \phi) \ dA = \int_{V} S_{\phi}(\phi) \ dV$$
(3)
$$\frac{\partial \rho_{p}}{\partial t} + \frac{\partial}{\partial x} (\rho \nu_{x}) + \frac{\partial}{\partial x} (\rho \nu_{r}) + \frac{\rho \nu_{r}}{r} = 0$$
(4)

2.2. Deep neural networks

Neural networks refer to broad type of non-linear models/parameterizations which involve combinations of matrix multiplications and other entrywise non-linear operations [20]. If the label (output variable; Y) has a linear relationship to the features (input variables; X), then a single neuron network is sufficient to predict the target (Eq. (5)). The term b and vector W is referred to as the bias and weight vector, respectively. It is interesting that the deep neural networks were inspired by biological neural network and each weight value correspond to the synapses [21]. A single neuron network also can be used for simple non-linear regression such as ReLU function as shown Eq. (6).

$$Y = W^{\mathsf{T}}X + b$$
, where $X \in \mathsf{R}^{\mathsf{d}}$, $Y \in \mathsf{R}$, $W \in \mathsf{R}^{\mathsf{d}}$ $b \in \mathsf{R}$ (5)
 $Y = \max(W^{\mathsf{T}}X + b, 0)$ $X \in \mathsf{R}^{\mathsf{d}}$, $Y \in \mathsf{R}$, $W \in \mathsf{R}^{\mathsf{d}}$ $b \in \mathsf{R}$ (6)

However, the machine learning technique generally aims to solve a problem with complex non-linear functions where multiple input variables are intertwined with each other [22]. It means that multi-layers composed of multiple neurons were frequently needed to predict the output variable according to the input variables. The deep neural networks can effectively follow complex non-linear functions by applying individual activation functions on each neuron [22].

In this paper, the theoretical background for construction of deep neural networks was explained based on simple two-layer neural networks. Fig. 1 shows the two-layer networks where I dimensional input variables and unit number of a hidden layer is J (Eq. (7)). Eq. (8) and (9) indicates the dimension of weight vector and bias corresponding to the dimension of the input variables and the hidden layer. W^1 is the first weight vector between the input layer and hidden layer, and W^2 is the second vector between the hidden layer and output layer. As different to the single neuron network, the number of parameters which were determined through training process increases with the neuron number of a hidden layer. Increasing the number of parameters can enhance the neural networks accuracy, but it is necessary to optimize the network size to prevent overfitting problems [23]. The downgraded performance of neural networks despite an increase in the number of parameters was also observed in this study. The details of the optimization process will be described in Section 4.1.

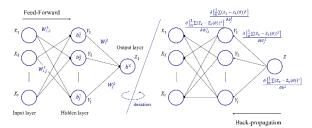


Figure 1. Schematic of fully connected layers. Feed-forward calculation for prediction and back propagation algorithm for parameter update.

$$X \in \mathbb{R}^{I}, Y \in \mathbb{R}^{J}, Z \in \mathbb{R}$$
 (7)
 $W^{1} \in \mathbb{R}^{I \times J}, W^{2} \in \mathbb{R}^{J},$ (8)
 $b^{1} \in \mathbb{R}^{J}, b^{2} \in \mathbb{R}$ (9)

The feed-forward algorithm is as follows, where the output variable is calculated by linear combinations (or non-linears with activation functions) consisting of trained parameters and input variables [24]. Each neuron in the hidden layer $Y_j \in \mathbb{R}$ is identical to a single neuron network as shown Eq. (10). Where

the dimension of each weight $W_{i,j}^1$, input variable X_i and bias b_i^1 are R.

$$Y_{i} = \sum_{i} W_{i,i}^{1} X_{i} + b_{i}^{1} \tag{10}$$

Subsequently, all values of hidden layer neurons are correlated in the form of linear function as shown **Eq. (11)**. Because it is written in the summation form as above, the matrix of variables is the same $(Z, W_j^2, Y_j, b \in \mathbb{R})$. When there is a non-linear variation of the output variable depending on the input variables, the activation function such as ReLU can be applied to capture the complex function as shown in **Eq. (12)**.

$$Z = \sum_{j} W_{j}^{2} Y_{j} + b^{2} = \sum_{j} \left(W_{j}^{2} \left(\sum_{i} W_{i,j}^{1} X_{i} + b_{j}^{1} \right) \right) + b^{2}$$
(11)

$$Z = \sum_{j} \left(W_{j}^{2} \cdot \text{relu}(Y_{j}) \right) + b^{2} = \sum_{j} \left(W_{j}^{2} \cdot \text{relu} \left(\sum_{i} W_{i,j}^{1} X_{i} + b_{j}^{1} \right) \right) + b^{2}$$
(12)

The summation expressions for neural networks in Eq. (10-12) can be simplified with matrix and vector notations as shown Eq. (13-15). Another important motivation of vectorization is the speed perspective in the training/testing calculation. In order to implement a neural network efficiently, vectorization takes advantage of matrix algebra and highly optimized numerical linear algebra packages to make neural network computations run quickly instead of using for loops (summation form) [25].

$$Y = (W^{1})^{\mathsf{T}}X + b^{1}$$
(13)

$$Z = (W^{2})^{\mathsf{T}}Y + b^{2} = (W^{2})^{\mathsf{T}}((W^{1})^{\mathsf{T}}X + b^{1}) + b^{2}$$
(14)

$$Z = (W^{2})^{\mathsf{T}} \cdot \operatorname{relu}(Y) + b^{2} = (W^{2})^{\mathsf{T}} \cdot \operatorname{relu}((W^{1})^{\mathsf{T}}X + b^{1}) + b^{2}$$
(15)

It was theoretically confirmed that that complex non-linear function can be predicted by deep neural networks, but to do that we need to appropriately determine the values of a large amounts of parameters. The back-propagation algorithm allows to optimize the parameter values of each neuron in deep neural networks [26]. **Eq.** (16) shows the mean square error (MSE) loss function (or cost function) for the n dataset $((X^{(1)}, Z^{(1)}), (X^{(2)}, Z^{(2)}), (X^{(k)}, Z^{(k)}), \cdots (X^{(n)}, Z^{(n)}))$. Where $Z^{(k)}$ is the ground truth value, $Z^{(k)}(\theta)$ is the predicted value by neural networks and θ means the set of all parameters (weight and bias).

$$J(\theta) = \frac{1}{\pi} \sum_{k=1}^{n} (Z^k - Z^k(\theta))^2$$
 (16)

The gradient descent (GD) optimizer, the most representative optimizer, allows to intuitively understand the procedure of the parameter optimization. In this method, parameters in deep neural networks can be updated in each iteration as shown Eq. (17). In addition, the stochastic gradient descent (SGD) optimizer was developed to improve the iteration speed of the GD optimizer. As shown Eq. (18), the SGD optimizer can update the parameters with sample *j* uniformly from *n* dataset.

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} J(\theta)$$

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} J^{(j)}(\theta)$$
(17)
(18)

Eq. (19) and (20) shows the more detailed parameter update procedure in the two-layer neural networks. It can be seen that direct gradient calculation is difficult in a layer which is not connected to the output layer in one step. For this reason, the chain rule is important to understand the back-propagation algorithm. The chain rule made it possible to solve the derivatives of composite functions in a very simple and elegant manner, simplifying the solution for very complex functions

$$W^{2} \leftarrow W^{2} - \alpha \left(\frac{\partial J}{\partial W^{2}}\right)^{\mathsf{T}}, b^{2} \leftarrow b^{2} - \alpha \frac{\partial J}{\partial b^{2}}$$

$$W^{1} \leftarrow W^{1} - \alpha \left(\frac{\partial J}{\partial W^{1}}\right)^{\mathsf{T}}, b^{1} \leftarrow b^{1} - \alpha \frac{\partial J}{\partial b^{1}}$$

$$(20)$$

$$W^1 \leftarrow W^1 - \alpha \left(\frac{\partial J}{\partial W^1}\right)^{\mathsf{T}}, b^1 \leftarrow b^1 - \alpha \frac{\partial J}{\partial b^1}$$
 (20)

For example, the gradient of loss function to the first weight vector $\frac{\partial J}{\partial w^1}$ can be calculated by the chain rule as shown Eq. (21). Likewise, in this study, multi-layer neural networks were used to imitate the non-linear governing equations in CFD and hence the parameters were determined by the back-propagation algorithm.

$$\frac{\partial J}{\partial W^{1}} = \left(\frac{\partial J}{\partial Y} \cdot \frac{\partial Y}{\partial W^{1}}\right)^{\mathsf{T}} = \left(\frac{\partial J}{\partial Y} \cdot X^{\mathsf{T}}\right)^{\mathsf{T}} = \left(W^{2} \cdot ((W^{2})^{\mathsf{T}} \cdot \mathrm{relu}((W^{1})^{\mathsf{T}}X + b^{1}) + b^{2} - Z\right) \cdot X^{\mathsf{T}})^{\mathsf{T}}$$
(21)

2.3. Development of the FVMN

In the previous sections, we understood the basic principles of CFD code with FVM (application), and machine learning with neural networks (tool). Although there have been recent studies attempting to predict the CFD time series data through the machine learning techniques [12, 28], characteristics of FVM were not sufficiently considered on network models. The neural networks with a general input/output system may accurately predict the data contained in the training set, but the accuracy can significantly reduce for blind test set. The increased discrepancy, between ground truth data (CFD) and predicted value by machine learning, with increase of time interval from the parameter update to the prediction time is

already a key issue [13]. For this reason, previous studies have been validated the neural networks model by limiting to the interpolated timeline data [12]. However, it is not only inefficient but also unfeasible to train the parameters in neural networks for all computational cases. It means that the development of a neural network model, which maintains reliable accuracy even in outside the trained timeline, is still necessary.

In this study, a finite volume method network (FVMN) was proposed considering the principles of FVM in the network input/output system. Fig. 2 shows the FVMN architecture by dividing into a training and prediction process. As shown in Fig. 2 (upper), in training process, the CFD time series data at time t and time t + 1 were used as input and output variables, respectively. The parameters (weight, bias) in neural networks were iteratively updated by the Adam optimizer with the backpropagation algorithm until the loss function converges.

The basic procedure is similar to the previous networks, but the tier and derivative system is applied in the FVMN In general neural networks, the matrix of input and output variables can be expressed in Eq. (22) and (23) (single input variable condition). Because $x_{i,j}$ represents the value of the variable at (i, j) grid position, the number of training examples is $n = M \times N$, where $M \times N$ are the number of grids in a 2D computational domain (Eq. (24)). The dimension of each input/output matrix is $R(=R^1)$ due to a single input variable.

$$\begin{split} X^{t} &= \left[x_{i,j}^{t} \right] \text{ where } X^{t} \in R \\ Z^{t} &= \left[x_{i,j}^{t+1} \right] \text{ where } X^{t+1} \in R \\ \left(\left(X^{(1)}, Z^{(1)} \right), \ \cdots \left(X^{(n)}, Z^{(n)} \right) \right) &= \left(\left(\left[x_{1,1}^{t} \right], \left[x_{1,1}^{t+1} \right] \right), \ \cdots \left(\left[x_{M,N}^{t} \right], \left[x_{M,N}^{t+1} \right] \right) \right) \end{split}$$

If there are I number of input variables to be considered (temperature, mixture composition, etc.), the matrix size is R^I as follows.

$$X^{t} = \left[x_{1_{t,j}}^{t}, x_{2_{t,j}}^{t}, \cdots x_{I_{t,j}}^{t}\right]^{\mathsf{T}} \text{ where } X^{t} \in R^{I}$$

$$\tag{25}$$

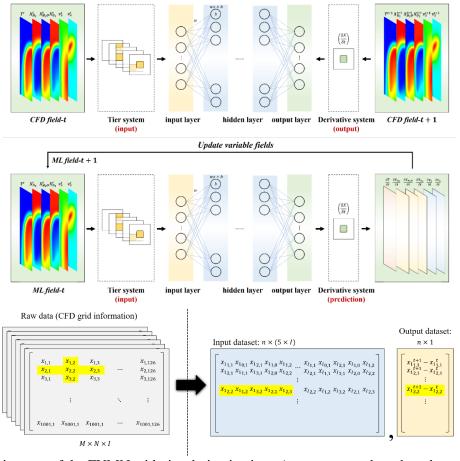


Figure 2. Architecture of the FVMN with tier-derivative input/output system based on deep neural network. (top) training procedure; (middle) prediction procedure; (bottom) preprocessing of CFD raw data for datasets.

On the contrary, in the FVMN model, the tier system was applied to the matrix of input variables. As shown in **Fig. 2** and **Eq. (26)**, a physical quantity in neighboring grids as well as the main grid was included in the input matrix. If there are I variable types to be considered, the matrix size is $R^{(5\times I)}$ as shown **Eq. (27)**. This tier input system can allow the neural networks to imitate the mass and heat transport between neighboring cells, which is considered the most important in FVM.

$$\begin{split} X_{t}^{t} &= \left[x_{i,j}^{t}, x_{i-1,j}^{t}, x_{i+1,j}^{t}, x_{i,j-1}^{t}, x_{i,j+1}^{t}\right]^{\mathsf{T}} \text{ where } X_{t}^{t} \in R^{\mathsf{5}} \quad (26) \\ X_{t}^{t} &= \left[x_{1_{i,j}}^{t}, x_{1_{i-1,j}}^{t}, x_{1_{i+1,j}}^{t}, x_{1_{i,j-1}}^{t}, x_{1_{i,j-1}}^{t}, x_{1_{i,j+1}}^{t}, \cdots x_{l_{i,j}}^{t}, x_{l_{i-1,j}}^{t}, x_{l_{i+1,j}}^{t}, x_{l_{i,j-1}}^{t}, x_{l_{i,j-1}}^{t}\right]^{\mathsf{T}}, X_{t}^{t} \in R^{(\mathsf{5} \times l)} \end{split}$$

Although neural networks with the tier system have been already investigated in previous research, the enhanced performance was not sufficient [12]. For this reason, we designed the derivative system in the output variables as shown Eq. (28). In our opinion, the derivative output variables can enhance the networks performance through scale separations. In

fluid analysis, the scale of the actual variable value of previous timestep is much larger than the differential to the next timestep. Because the time series data in next timestep can be updated by solely predicting the change amount Z_d^t , adding the actual value with a much higher scale in the output variable $Z^t = Z^{t-1} + Z_d^t$ may impair the networks performance. It should be noted that, in various neural networks field, there have been successful attempts to improve the performance through the scale separation [10]. **Fig. 2** (bottom) shows the preprocessing of CFD raw data for application of the tier-derivative system in the FVMN model. A three-dimensional raw data matrix is transformed into two-dimensional matrixes for input and output datasets.

In conclusion, the neural network should aim to predict the amount of change in a physical quantity $\frac{x_{i,j}^{t+1}-x_{i,j}^t}{\delta t}$ over by all physical quantities X_t^t in neighboring grids as well as the main grid (**Fig. 2** (**lower**)). Since individual networks according to

each output variable was constructed in this study, the dimension of the output variables matrix is R regardless of the number of output variables (Eq. (29)). The CFD time series data at t + 1 was finally predicted by the sum of the data at t and the predicted derivative amount $X^{t+1} = X^t + \delta t \cdot Z_d^t$. By this procedure, the continuous prediction of the CFD time series data is possible.

$$\begin{split} Z_{d}^{t} &= \left[\left(\frac{\delta x}{\delta t} \right)_{i,j}^{t+1} \right] \text{ where } Z_{d}^{t} \in R \\ Z_{d}^{t} &= \left[\left(\frac{\delta x_{1}}{\delta t} \right)_{i,j}^{t+1} \right] \text{ where } Z_{d}^{t} \in R \end{split} \tag{28}$$

$$Z_d^t = \left[\left(\frac{\delta x_1}{\delta t} \right)_{i,j}^{t+1} \right] \text{ where } Z_d^t \in R$$
 (29)

The performance of machine learning with the general network model and FVMN model was comprehensively compared by divided into four cases (general, with tier system, with derivative system and FVMN) in Section 4.2.

2.4. Test matrix

The optimization of the hyperparameters in neural networks has become an essential step since the overfitting problem was emerged. As the model complexity increases, the training loss and test loss generally decrease together in underfitting level of deep learning. The test dataset is not included when determining the parameters of the neural networks (i.e. training step). From a certain complexity level, the test loss starts to rebound and increases, unlike the training loss. This is the overfitting problem, and in this case it is difficult to assure the model accuracy for the untrained datasets. Because the FVMN aims to accurately predict the multistep CFD time series data, the optimization of the neural networks is necessary to prevent the overfitting issue.

Table S1 shows the test matrix for the optimization of the FVMN in this study. In the optimization step, several hyperparameters which cannot change during the training step are determined. The number of hidden layers and neurons, activation function and learning rate are typical hyperparameters. In case (a-d), the relationship between the number of hidden layers and model accuracy was investigated. In case (c) and (e), the activation function performance of ReLU and Sigmoid was compared. In case (c, f, g), the sensitivity of the neuron number of hidden layers was investigated. Finally, in case (h), the neural networks performance in the funneltype architecture was explored with reference to Ref. [29]. In this study, the learning rate was fixed at 0.001 in all cases. Like practical Bayesian optimization method, there are more extensive

optimization methods determine to the hyperparameters [30]. Since the main objective of this study is to confirm the feasibility of the FVMN in CFD application, the hyperparameters are determined by the sensitivity study with case (a-h). As shown in **Table S1**, the number of parameters varies greatly on a scale of 1,000 to 100,000 depending in the network architecture. In addition, MSE for the validation dataset was used as the loss function to determine the parameters in training step.

3. DATASETS

3.1. CFD simulation

To optimize and evaluate the performance of the FVMN model, reliable CFD simulation results are required. In this study, the datasets were chosen from time series data from a reacting flow simulation which is still less active subject in machine learning applications. Unlike general fluid analysis, reacting flow analysis includes an elaborate calculation of gas species diffusion and reaction behavior. As mentioned in Section 1, the soar of computational cost in the reacting flow emphasizes the necessity of accelerating the calculation speed through the application of machine learning technique.

Recently, Jeon et al. investigated the hydrogen flame extinction process through computational flame analysis [7]. They numerically observe the steady hydrogen flame by the stabilized flame method. The stabilized flame method has advantage to overcome the difficulty of relying on optical observation to measure the limit concentration in the standard flammability tube apparatus. In their work, suitable CFD models and grid size to simulate the flame propagation of lean hydrogen flames were identified [7].

Therefore, we produced training/validation/test datasets for the FVMN by the stabilized flame simulation referring the previous works. The simulated flame has a 5% lean hydrogen mixture condition with pure air. Fig. 3 (a) shows the axisymmetric computational domain for stabilized flame simulation and Fig. 3 (b) shows the flame peak temperatures were converged at 0.2 mm grid size. A uniformly structured mesh $(0.1 \text{ mm} \times 0.1 \text{ mm})$ mm) was adopted in our simulation (M =1001, N = 126). Jeon et al. suggested that the coarse grid size cannot calculate the continuous combustion process by coupling gas transport and chemical kinetics. The discrete ordinates (DO) radiation model calculates the thermal radiation for a finite number of discrete solid angles. Due to the very thin flame thickness, flames of limiting mixtures are close to the optical thin condition. The DO radiation model is known to cover the broad range of optical thicknesses. In gas combustion simulation, the calculation of the absorption coefficient in the domain is required. For this reason, the absorption coefficient of each cell was calculated by using the weighted-sum-of-graygases (WSGG) model. The model is known to a compromise between the oversimplified gray gas model and the complete model [31].

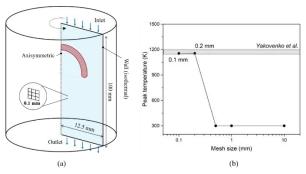


Figure 3. Axisymmetric cylindrical domain and boundary conditions (a) and grid sensitivity analysis results based on peak flame temperature (b) [7].

The non-slip and isothermal wall conditions were adopted due to a observed slight increased wall temperature in the experiments [32]. A negligible effect of the slightly elevated wall temperature was confirmed by the sensitivity studies [7]. The detailed descriptions about the grid sensitivity and the models can be found in Ref. [7]. The transient solver was used since the goal of the FVMN model is the prediction of the CFD time series data. The timestep for transient analysis was selected as 1 ms by referring to the sensitivity results of the previous study [33]. To further validate our unsteady flame simulation, the results were compared with the steady simulation under the similar mixture condition in Fig. S1. The flame temperature and hydrogen mole fraction in the centerline shows good agreement between the two results.

Understanding the governing equation used in our CFD simulation is essential to figure out the relationship between input and output variables. Since the calculation process of the governing equations combined with the models was described in detail in Ref. [7], representative equations confirming the non-linearity of variables were introduced in this study. Equations (30) and (31) show the continuity and momentum equation in governing equations for unsteady respectively.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{30}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0$$

$$\frac{\partial}{\partial t} (\rho \vec{v}) + \nabla \cdot (\rho \vec{v} \vec{v}) = -\nabla p + \nabla \cdot (\bar{\tau}) + \rho \vec{g}$$
(30)

 $\bar{\tau}$ is the stress tensor can be calculated by the viscous model (Eq. (32)) since the time-averaged Reynolds stress tensor does not need to be calculated in laminar conditions. μ is the molecular viscosity and I is the unit tensor. The non-linearity between variables can be identified through the term $\rho \vec{v} \vec{v}$. In addition, the neighboring cell information is necessary to calculate the gradient of gas velocity $\nabla \vec{v}$.

$$\bar{\bar{\tau}} = \mu \left[(\nabla \vec{v} + \nabla \vec{v}^T) - \frac{2}{3} \nabla \cdot \vec{v} I \right]$$
 (32)

In our simulation, the species transport model, which solves a conservation equation describing convection, diffusion, and detailed chemical kinetics for each component species, was used to calculate the flame behavior in the micro-region as shown in Eq. (33). To calculate the net rate of production for each species R_i , the rate constants are computed using the Arrhenius equation as shown in **Eq. (34)** [34]. The San Diego mechanism was modelled to calculate the sub-chemical kinetics of hydrogen-oxygen combustion by user define function (UDF). The mechanism consists of 20 reversible elementary reactions with 8 reactive species including radicals. $\vec{J_l}$ is the diffusion flux of each species caused by gradients of concentration Y_i (neglect of Soret effect). The diffusion flux of each species was calculated as the product of density and the concentration gradient. Since density can be expressed as a function of temperature in ideal gas model, the term leads to complex non-linear calculations. The non-linear term was also included in the energy equation as shown in **Eqs.** (33) and (34).

$$\frac{\partial}{\partial t}(\rho Y_i) + \nabla \cdot (\rho \vec{v} Y_i) = -\nabla \cdot \vec{J}_i + R_i$$

$$K = AT^b \exp\left(-\frac{E_a}{RT}\right)$$

$$\vec{J}_i = -\rho D_{i,m} \nabla Y_i$$
(33)

$$K = AT^b \exp\left(-\frac{E_a}{pT}\right) \tag{34}$$

$$\vec{J_i} = -\rho D_{i,m} \nabla Y_i \tag{35}$$

For gas combustion analysis, the energy equation should consider the reaction heat, radiation, and species diffusion. As shown in Eqs. (33) and (34), the non-linear term was also included in the energy equation.

$$\nabla \cdot (\vec{v}(\rho E + p)) = \nabla \cdot (k\nabla T - \sum_{i} h_{i} \vec{J}_{i}) + S_{h}$$
 (36)

$$E = h - \frac{p}{\rho} + \frac{v^2}{2} \tag{37}$$

The need for deep neural network to deal with the non-linearity and the tier system, introduced in *Section 2*, was highlighted in the governing equations. In addition, the required input variable types for prediction of time series data were confirmed. To compute the continuity and momentum equation, the temperature and velocity (axial and radial velocity) variables of all grids are required. A negligible local pressure increases was observed due to the characteristics of the lean limit flame, so the pressure variable was not included in datasets. Each species concentration is also needed to calculate the diffusion flux in the energy and species conservation equation.

3.2. Datasets

The datasets for this study consisted of a partial timeline within the entire stabilized flame simulation timeline. The entire timeline of the stabilized flame generation process, numerically calculated in this study, was depicted in **Fig. 4**. At 0 s before ignition, the temperature of all grids was 300 K. In 0.05s, the temperature and flame propagate out of the ignition area by the occurrence of ignition. The flame continues to expand until 0.5 s when the ignition energy is in effect. After 0.5 s, the flame begins to stabilize through the balance of heat loss mechanisms (conduction, radiation, convection) and combustion heat generated by the existing flame. This flame stabilizing period was selected as the subject of this study, hence the simulation results on the 0.600-0.611 time series was used as datasets. Specifically, the results on 0.600-0.601 were used as training/validation sets (0.8/0.2) and 0.601-0.611 were used as test sets.

As discussed in *Section 3.1*, input variables used to training machine was consists of flow variables v_x^t , v_r^t and temperature variables T^t , and species variables $X_{H_2}^t$, $X_{H_20}^t$, $X_{O_2}^t$. Unlike previous studies [28], the grid location information was not included in the input variables to confirm the universal feasibility of FVMN model, which is not restricted to the domain geometry. The concentrations of radicals, such as H and O, were not included in the input/output variables because they are scaled close to zero even in the reaction zone. Investigation of effect of the inclusion of radical variables on loss function is our future work.

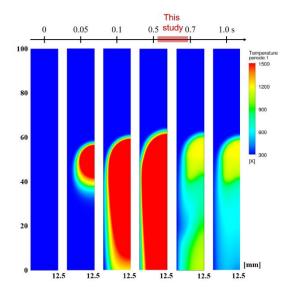


Figure 4. Overall stabilized flame generation process observed by the CFD simulation.

In data driven research, the feature scaling to bring all features in the similar standing is necessary so that the effect of the specific feature cannot be biased. Fig. 5 (a) shows that the original input variables have a large scale difference according to the type. Specifically, the scale difference between the temperature at the 10^3 scale and the hydrogen mass fraction at the scale 10^{-3} is significantly noticeable. For this reason, the standardization of input variables which is representative scaling technique where the values are centered around the mean with a unit standard deviation (Eq. (38)) was conducted in this study. It was confirmed that the ranges of the standardized input variables are quite similar as shown Fig. 5 (b). Fig. S2 shows the two dimensional standardized data distribution in the training/validation examples based temperature and axial velocity.

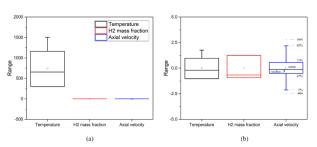


Figure 5. Boxplot of input variables before (a) and after standardization (b). The apparent difference in scale of variables was resolved.

$$X_t = \frac{X_t^{raw} - \mu}{\sigma} \tag{38}$$

The six individual networks were trained to predict each derivative amount of output variable $(\frac{v_x^{t+1}-v_x^t}{\delta t}, \frac{v_r^{t+1}-v_r^t}{\delta t}, \frac{X_{H_2}^{t+1}-X_{H_2}^t}{\delta t}, \frac{X_{H_2}^{t+1}-X_{H_2}^t}{\delta t}, \frac{X_{H_2}^{t+1}-X_{H_2}^t}{\delta t}, \frac{X_{O_2}^{t+1}-X_{O_2}^t}{\delta t}, \frac{X_{O_2}^{t+1}-X_$

3.3. Machine learning domain

As shown **Fig. 6**, the computational domain was numerically solved by being divided into a machine learning domain and a CFD domain. In the CFD domain, the time series data in the next timestep are calculated by solving the first principles (governing equations discussed in Section 3.1). Only in the machine learning domain, the time series data are calculated by the FVMN model. The dimension of original data matrix in the entire domain A (before data preprocessing for machine learning) is $R^{M,N}$ where M is the number of grids in the axial direction and N is the number of grids in the radial direction. As shown Eq. (39), the domain for M^* layers in both inlet A_{inlet} and outlet region A_{outlet} were classified into the CFD domain ($M^* = 100$). The rest of domain excluding the CFD domain (inlet/outlet region) are designated as the machine learning domain $A_{flame} \in \mathbb{R}^{M-2M^*,N}$. In other words, the input and output variables for FVMN are extracted from the machine learning domain A_{flame} as shown Eq. (40). This is equivalent to saying that the number of training examples is n = 801×126 .

There are two reasons for suggesting the combined ML-CFD calculation method in this study. First, the computational domain selected in this study has a height of 100 mm. It means that the distance between the main flame calculation region and the inlet/outlet is quite short. For this reason, the variation of the time series data in the inlet and outlet regions has stiffness in the numerical solution. In our preliminary analysis, it was observed that this stiffness can significantly limit the application of data-driven methods. Second, as introduced in the Section 1, a sharp decrease in accuracy was identified as the timestep passed in many ML-CFD application studies. These limitations may be caused by the neural networks not being able to

imitate the boundary condition of CFD simulation. We investigated whether these intrinsic limitations of data-driven methods can be alleviated in the combined calculation method.

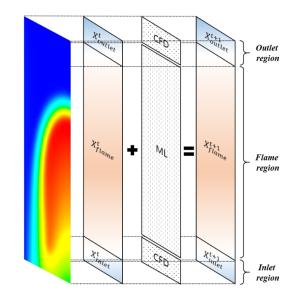


Figure 6. Schematic of CFD coupled ML method. The computational domain is divided into three regions, of which input/output region and flame region is updated by CFD and ML respectively.

$$A^{t} \in \mathbb{R}^{M,N}, A^{t}_{inlet} \in \mathbb{R}^{M^{*},N}, A^{t}_{outlet} \in \mathbb{R}^{M^{*},N}, A^{t}_{flame} \in \mathbb{R}^{M-2M^{*},N}$$

$$X^{t}_{t} \subset A^{t}_{flame}, Z^{t+1}_{d} \subset A^{t+1}_{flame} \text{ where } X^{t}_{t} \in \mathbb{R}^{(5 \times I)}, Z^{t+1}_{d} \in \mathbb{R}$$

$$(39)$$

3.4. Boundary conditions

As discussed in *Section 2.3*, the tier system which includes information on neighboring grids as input variables was applied to FVMN model. There are remained boundary conditions for walls in our CFD simulation (**Fig. 7**) except for inlet and outlet. For a center domain where the main grid is located in the second or more grid layer, the input variables can be figured in the normal tier system as shown **Eq. (41)**. On the contrary, the boundary layer (first grid layer) lacks a grid information beyond the wall. If the dimension of input variable matrix varies depending on the grid position, it will lead to a training loss increasement.

In the axisymmetry wall, this component shortage problem can be handled similarly to the method solving the zero Neumann boundary condition in finite difference method. As shown **Fig. 7** and **Eq.** (42), the variable information of the main grid can replace the grid beyond the wall. Since a rest wall is in isothermal conditions, wall information is required as shown **Eq.** (43). To apply wall

information, the temperature variable should be set to the wall temperature, and other variables (species concentrations and axial, radial velocity) set to a value $x_{1,wall} = T_{wall}, x_{2,wall} = x_{3,wall} =$ $x_{4,wall} = x_{5,wall} = x_{6,wall} = 0$. However, it was observed that the application of wall condition induces higher error compared to the zero Neumann boundary condition as shown Fig. S3. When wall information is included in the wall boundary domain, a discrepancy of scales of input values with major domain emerges and hence training error is Because the training examples magnified. containing the wall information is a small portion of the entire examples, it seems difficult for neural networks to distinguish this dataset in the training process. Therefore, in this study, component shortage problem in the wall was also handled by the symmetry condition (Eq. (44)).

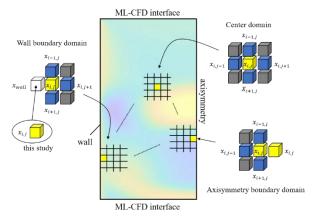


Figure 7. Schematic of machine learning input datasets in boundary domain for tier system.

All following results in *Section 4*, the prediction of the next CFD time series data using FVMN is the results of applying the standardization of the input variables, domain partitioning and zero Neumann boundary condition described in this section.

$$\begin{aligned} &(Z_{d}^{t+1})_{center} = f\left(\sum_{k=1}^{6} x_{k_{i,j}}^{t}, x_{k_{i-1,j}}^{t}, x_{k_{i+1,j}}^{t}, x_{k_{i,j-1}}^{t}, x_{k_{i,j+1}}^{t}\right) \text{ where } & x_{k}^{t} \in X_{t}^{t} \\ &(2l) \\ &(Z_{d}^{t+1})_{axis} = f\left(\sum_{k=1}^{6} x_{k_{i,j}}^{t}, x_{k_{i-1,j}}^{t}, x_{k_{i+1,j}}^{t}, x_{k_{i,j-1}}^{t}, x_{k_{i,j}}^{t}\right) \text{ where } & x_{k}^{t} \in X_{t}^{t} \\ &(42) \\ &(Z_{d}^{t+1})_{wall} = f\left(\sum_{k=1}^{6} x_{k_{i,j}}^{t}, x_{k_{i-1,j}}^{t}, x_{k_{i+1,j}}^{t}, x_{kwall}, x_{k_{i,j+1}}^{t}\right) \text{ where } & x_{k}^{t} \in X_{t}^{t} \\ &(43) \\ &(Z_{d}^{t+1})_{wall} = f\left(\sum_{k=1}^{6} x_{k_{i,j}}^{t}, x_{k_{i-1,j}}^{t}, x_{k_{i+1,j}}^{t}, x_{k_{i,j}}^{t}, x_{k_{i,j+1}}^{t}\right) \text{ where } & x_{k}^{t} \in X_{t}^{t} \end{aligned}$$

4. RESULTS AND DISCUSSIONS

4.1. Optimization of hyperparameters

As described in Section 2.2, the degraded performance of neural networks despite an increase in the networks size (overfitting problem) is common issue in the machine learning industries. For this reason, it is necessary to optimize the number of layers and nodes which are considered the most important hyperparameters. In this study, optimization was conducted by comprehensively evaluating the errors of the trained FVMN for training/validation dataset. More specifically, the neural networks were trained with 80% of 0.600-0.601 time series data as shown Fig. 2 (top). For each epoch, MSE for the remaining 20% data was calculated and the training process ends when the loss function converges. After training, the trained networks predicted all 0.601 time series data by 0.600 time series data as shown Fig. 2 (middle). The information about the network structure in each case is described in Table S1.

Fig. 8 shows the error distribution between the ground truth data (CFD) and the predicted value (relative error for each grid), based on the temperature value at 0.601 s. The color scale is the same for all cases (0 - 0.05%). In case (a) with the simplest network structure, high errors were observed in most of the interface region between the flame and unburned gas due to the underfitted networks. In cases (c) and (d), the underfitting problem was gradually alleviated by increasing the number of hidden layers. However, when the number of hidden layers increased to 4 (case (d)), overall higher errors were observed than in case (c). The pronounced error regions were the flame concave region and the outlet region, where the variable change amount was focused. It means that the overfitted networks caused soar of errors in the stiff calculation region.

This overfitting phenomenon was also confirmed for the number of neurons per hidden layer as shown cases (f) and (g). Although the number of parameters is required more than 10 times compared to case (c), the errors in the concave region are higher. The funnel-shaped structure which have been widely used in the field of image learning cannot make noticeable improvements in accuracy. In case (e) using sigmoid activation function, the maximum and mean relative error is higher than ReLU function case.

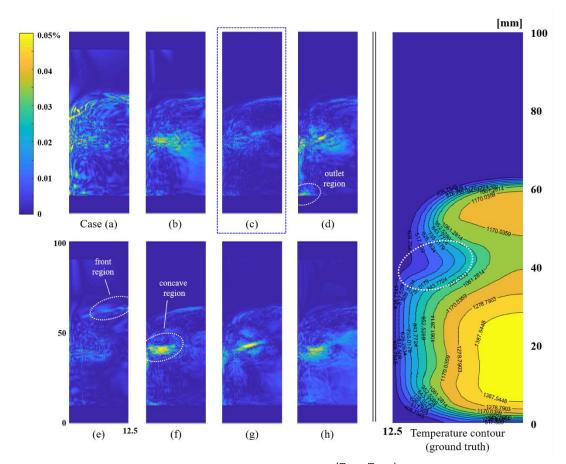


Figure 8. Relative error distribution of predicted temperature $\frac{|T_{ML}-T_{CFD}|}{T_{CFD}}$ based on the CFD results of 0.601 s. The properties of each fully connected layer are shown in **Table 1**.

In addition to the comprehensive comparison with the entire error distribution, the maximum and mean relative error in each case were quantitatively compared in **Fig. S4**. In both error graphs, the underfitting and overfitting level can be more clearly identified. As a result, the FVMN was optimized in the structure of 3 hidden layers and 64 nodes per hidden layer (case (c)). It should be noted that the structural optimization results were also verified for the test dataset (0.602-0.611 time series data). The following results were based on the optimized network structure.

4.2. Performance of the FVMN compared to the general network models

In this study, the FVMN with the tier-input and derivative-output system was proposed to predict the CFD time series data more accurately. The new concept which mimics the principles of CFD calculation procedure has the advantage of being

intuitive, but it should be investigated whether this concept can actually improve the performance. **Fig. 9** shows the effects of the tier and derivative system application on the error reduction based on the temperature value at 0.601 s. For all cases, the hyperparameters including the networks size are the identical as those optimized in *Section 4.1*. The only difference in case (1-4) is the form of input and output variables during training and prediction process. In case (1) with FVNM, as discussed in *Section 2.3*, the input and output variable matrix was preprocessed by the tier and derivative system as shown **Eq. (45)** and **(46)**.

$$\begin{split} X_{t}^{t} &= \left[x_{1,j}^{t}, x_{1_{t-1,j}}^{t}, x_{1_{t+1,j}}^{t}, x_{1_{t,j-1}}^{t}, x_{1_{t,j-1}}^{t}, x_{1_{t,j+1}}^{t}, \cdots x_{l_{t,j}}^{t}, x_{l_{t-1,j}}^{t}, x_{l_{t+1,j}}^{t}, x_{l_{t,j-1}}^{t}, x_{l_{t,j-1}}^{t}\right]^{\mathsf{T}}, X_{t}^{t} \in R^{(5 \times l)} \\ Z_{d}^{t} &= \left[\left(\frac{\delta x_{1}}{\delta t}\right)_{l,j}^{t+1}\right] \text{ where } Z_{d}^{t+1} \in R \end{split} \tag{46}$$

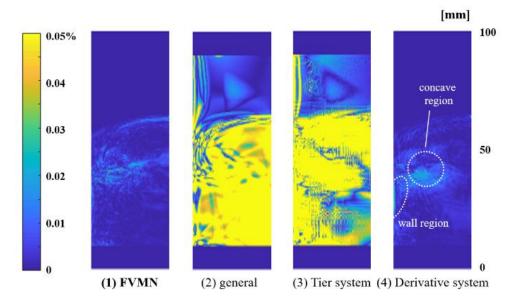


Figure 9. Relative error distribution of predicted temperature $\frac{|T_{ML}-T_{CFD}|}{T_{CFD}}$ depending on the application of the tier and derivative system (based on the CFD results of 0.601 s).

For case (2) with general network model, only the actual variable information of the main grid is used as the input and output variable shown in Eq. (47) and (48).

$$X^{t} = \begin{bmatrix} x_{1_{i,j}}^{t} \cdots x_{l_{i,j}}^{t} \end{bmatrix}^{\mathsf{T}}, X^{t} \in R^{(I)}$$

$$Z^{t} = \begin{bmatrix} x_{1_{i,j}}^{t+1} \end{bmatrix} \text{ where } Z^{t} \in R$$

$$(48)$$

$$Z^{t} = \left[x_{1_{i,j}}^{t+1} \right] \text{ where } Z^{t} \in R \tag{48}$$

In case (3) and (4), each effect on the performance of the tier and derivative system was investigated. Eq. (49) and (50) shows the network model with only tier system, while Eq. (51) and (52) shows the model with only derivative system.

$$\begin{split} X_{t}^{t} &= \\ \left[x_{1_{i,j}}^{t}, x_{1_{i-1,j}}^{t}, x_{1_{i+1,j}}^{t}, x_{1_{i,j-1}}^{t}, x_{1_{i,j-1}}^{t}, \cdots x_{l_{i,j}}^{t}, x_{l_{i-1,j}}^{t}, x_{l_{i+1,j}}^{t}, x_{l_{i,j-1}}^{t}, x_{l_{i,j+1}}^{t}\right]^{\mathsf{T}}, X_{t}^{t} \in R^{(5\times l)} \\ Z^{t} &= \left[x_{1_{i,j}}^{t+1}\right] \text{ where } Z^{t} \in R \\ X^{t} &= \left[x_{1_{i,j}}^{t+1}\right] \mathbf{v}, X^{t} \in R^{(l)} \\ Z_{d}^{t} &= \left[\left(\frac{\delta x_{1}}{\delta t}\right)_{i,j}^{t+1}\right] \mathbf{v}, X^{t} \in R^{(l)} \end{aligned} \tag{51}$$

As a result, it was noted that the significantly improved performance of the FVMN approach were confirmed compared to the general model as shown Fig. 9. In case (2), the relative error with the CFD data exceed 0.05% in not only the interface region between the flame and unburned gas but also almost all regions with burned gas. The maximum relative error of 1.12% is much higher than the FVMN of 0.04%. The performance of the general model is worse than any of the investigated networks in the optimization study in Section 4.1.

In case (3), it was observed that errors in some grids are alleviated by applying the tier system (maximum relative error of 0.26%).

The interesting is that the efficacy of the derivative system is much more noticeable than that of the tier system, although previous studies have only emphasized the importance of the tier system [12, 28]. As shown case (4), the application of the derivative system significantly reduced the errors even in the stiff calculation region. As discussed in Section 2.3, the scale of the actual variable value of previous timestep is much larger than the differential to the next timestep. For example, the scale of temperature values reaches about 100 times the scale of the amount of temperature change in these cases. We concluded that the scale separation of output variables by the derivative system has a significant effect improving on networks performance. These results highlighted that the derivative-output system is essential to utilize the deep learning technique to CFD application.

Although the overall error distribution is similar between cases (1) and (4), but the local error at the wall boundary region and the flame concave region is higher in case (4). If only the main grid information is included in the input variables without the tier system, it is difficult for networks to identify the wall grid by itself. As we expected, the tier system can improve the networks performance of calculating the transport of physical quantities between grids. In conclusion, the concept of the FVMN which mimics the principles of CFD calculation procedure has explicit efficacy for predicting time series data. In the next section, the FVMN performance for multi-step prediction will be investigated.

4.3. Performance of the FVMN in multi-step prediction

The ultimate goal of the FVMN is to provide feasibility of costly reacting flow simulations by accelerating CFD computation. It means that the trained networks should be able to precisely predict multi-step CFD time series data which is not included in the training dataset. In this study, we evaluated the performance of the FVMN in multistep prediction by using our CFD data (Section 3.1). The single-step time series data (0.600 - 0.601 s)from the CFD simulation was used as training dataset and the multi-step data (0.601 - 0.611 s) was used as prediction datasets (i.e. test datasets). If using the multi-step training dataset, it may be possible to maintain the networks performance in broader time range, but a larger networks size and training time are necessarily required. Investigation of the change in networks performance according to the training set size is our future works.

Fig. 10 shows the variation of FVMN performance according to the progress of the timestep (bluerectangular line). The maximum relative error (filled symbol) and the mean relative error (hollow symbol) show almost same trend. For the trained time series data at 0.601 s, the maximum relative error of 0.04% was identified and it has much smaller mean relative error (0.002%). In the first prediction dataset (0.602 s), the maximum error is 0.10% and the mean error is 0.006%. From the prediction dataset, output variables are calculated as the sum of the ML time series data in the previous timestep and the variable gradient predicted by the ML data $X_{ML}^{t+1} = X_{ML}^t + Z_d^t(\sum X_{t,ML}^t)$. In other words, in ML domain, network model is completely independent on CFD time series data (ground truth value) because the CFD data is no longer used to predict the time series data. In next prediction dataset (0.603 s), the maximum error is 0.21% and the mean error is 0.015%. It was confirmed that both errors in the prediction datasets increase in the form of quadradic function for timestep. In the last dataset (0.611 s), the maximum error reaches about 2.6%. Although the mean error level (0.24%) seems to be acceptable in CFD analysis, this trend of error increasement cannot be neglected for longer timestep analysis.

There are two categories that can account for the discrepancy between the CFD time series data and ML time series data, the initial condition error and the gradient error. Because the output variables were calculated based on the previous ML time series data instead of CFD time series data, the error which occurred in the previous calculation is already included in the initial condition. In addition, there is the gradient error which occurs in the network performance itself predicting the gradient by input variables. We observed these errors during the single-step prediction case to distinguish the impact of each error. The single-step prediction case means that the gradient amount in each timestep is calculated by the CFD time series data $X_{ML}^{t+1} = X_{CFD}^t + Z_d^t(\sum X_{t,CFD}^t)$. By removing the initial condition error, only the magnitude of the gradient error can be investigated. As shown in black-circular lines, the maximum and mean gradient error increases linearly with timestep. In this case, the maximum error does not exceed 0.5% even at the last timestep. Therefore, it was confirmed that the error accumulation in multi-step prediction is mainly caused by the initial condition errors. The linearly increasing error observed in the single-prediction case was accumulated, resulting in the quadratic error function in the multi-step prediction case.

If we can reduce the gradient error through additional optimization of FVMN or advance of loss function, acceleration of CFD calculation over a wider time range will become feasible. However, we considered this linear error increasement itself is an inevitable phenomenon which occurs as the training point and the prediction point become distant. The solution about this linear increase of the gradient error will be discussed in more detail in *Section 4.4*.

In addition, the red-triangle lines represent the error accumulation when the gradient of each variable obtained from 0.600-0.601 s data is constantly applied to the whole test dataset. Although the small size of each timestep (0.001~s) and the small number of timesteps (10~times), the error increase much faster than in the cases of using FVMN. In the 4th prediction dataset, the maximum error already exceeds 2%. These results highlighted that the sequential gradient prediction by machine learning technique is necessary to accelerate the CFD simulation.

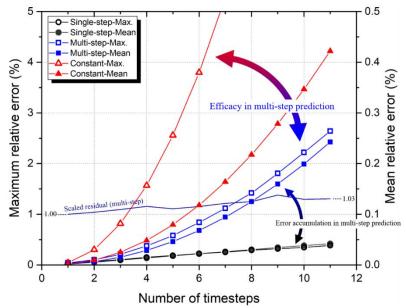


Figure 10. Changes in maximum and mean relative errors in single-step prediction, multi-step prediction, constant gradient cases. The feasibility of the multi-step prediction with FVMN were confirmed, and the concept of Machine learning aided CFD network was concurrently proposed.

Fig. 11 shows the relative error distribution as the timestep passed. Observing the error distribution in ML domain makes it possible to infer the main reason of the gradient error, which was difficult in the maximum error analysis. As discussed above, in multi-step prediction case (Fig. 11 (a)), the error gradually increases over time. In the time series data at 0.604 s, the flame front was depicted together with the error distribution. It was confirmed that the local error generally increased around the flame front, especially in the region near the concave front. The error generated in the concave region propagated to the vicinity as the ML prediction continues. Although the magnitude of the relative error is difference, this error propagation pattern is very similar in the single-step prediction case as shown **Fig. 11 (b)**.

The first reason the error is noticeable in the concave region is that the gradient size itself is larger than other regions. As the value of the gradient to be predicted increases, the relative error with respect to the actual value has a disadvantage. The second reason seems to be the limited ratio of the concave region data included in the training dataset.

Fig. 12 shows the distribution of the output variables Z_d^{t+1} based on the training/validation dataset. There is a clear difference from the normal distribution, which is generally encouraged to improve the training effectiveness in machine learning techniques. The biased distribution of

dataset contained output variables ranging from -2.5 to 2.5 K making up about 60% of the total data. However, the concave region where a relatively large error occurs includes a higher Δt region as shown in **Fig. 12**. Consequently, the networks performance can be degraded in the specific region when the networks was not sufficiently trained for partial data having a relatively small distribution. If an advanced sampling method is used in training process, the gradient error can be alleviated. Since the aim of this study is to develop a baseline FVM network model, this is our future work.

In addition, we compared the FVMN time to the time the CFD solver takes to drop the tolerable residual scale. Although neural performance is more advantageous for GPU sever, an Intel Core i7-6700 CPU (TensorFlow 2.3.0) was used in both calculations for direct comparison. It was confirmed that the average time for calculation of next timestep data was about 10 times faster in FVMN. This reduction in computational cost will be more pronounced when GPU server is used. Once the number of predictable multi-steps by the single training can be determined, more accurate comparisons including the training time become feasible.

4.4. Machine learning aided CFD framework

In the previous section, the mean relative error of 0.24% was identified when predicting CFD time

series data for 10 timesteps after training the networks with only a single timestep. The improved accuracy is attributed to the characteristic of the FVMN which thoroughly reflect the CFD principles. This mean error level can be considered an acceptable error, but the linearly increasing gradient

error is a remaining issue in transient calculations for longer times. In other words, although an advanced network model can reduce the increase rate, this linear error increase itself seems to be an inevitable phenomenon in data driven methods.

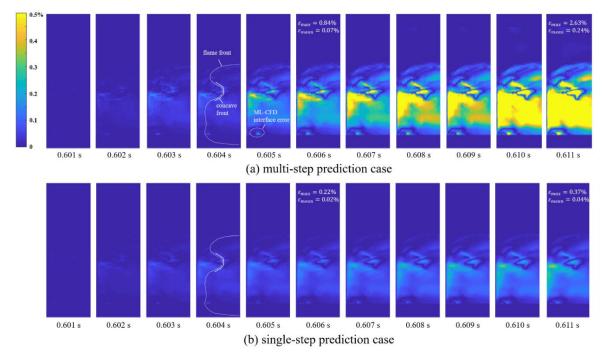


Figure 11. Relative error distribution of predicted temperature $\frac{|T_{ML}-T_{CFD}|}{T_{CFD}}$ over timestep. The error is accumulated in the multi-step case unlike the single case where the error increase linearly.

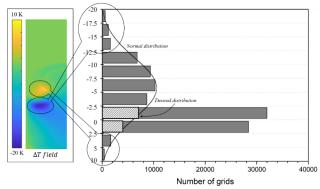


Figure 12. Distribution of the output variable $Z_d^{t+1} = \left[\left(\frac{\delta x}{\delta t} \right)_{i,j}^{t+1} \right]$ based on the CFD time series data 0.600 - 0.601 s. The biased data distribution may cause the pronounced error in the stiff region.

Therefore, we suggested a MACnet framework— a Machine learning aided CFD framework which can accelerate the CFD simulation through alternating computations (**Fig. 13**). As the first step. the CFD calculation for a single step is conducted by solving

governing equations. The number of timesteps included in the training dataset may depend on the complexity of each simulation. In the next step, the calculated CFD time series data train the constructed neural networks to determine the parameters. Then, the trained neural networks can quickly predict the next time series data without costly CFD simulation. Up to this point, it is the identical as the multi-step prediction case we covered in the previous section. However, the MACnet framework reverts back to the CFD calculation when the ML computation error is close to the tolerance. This method seems to be feasible because there is an acceptable residual range even in the CFD simulation itself. When the conservation error converges below a certain level in a timestep, the calculation of the next timestep starts in unsteady CFD simulations. The new CFD results in MACnet is used for training dataset to again update the parameters of neural networks. The evolved neural networks again accelerate the CFD simulation by calculating the multi-step time series data, and this process is repeated until the end of the calculation. The key advantage of the MACnet is that it can be applied individually to each simulation. Although some optimization works of networks will be required depending on the simulation complexity, there is no need to train the networks by the enormous amount of simulation results.

In order for the MACnet to accurately predict the CFD time series data for a long time, the trend of error variation in each ML timestep should be appropriately monitored. For the flow field to be predicted, there is no ground truth data (CFD result) unlike in the training dataset. It means that the relative error compared to the ground truth data cannot be checked periodically to determine whether or not the tolerance is exceeded. We considered that the residual value calculated in the similar manner with the CFD simulation could be provided as a monitoring variable. This idea stems from recent studies trying to enhance networks performance by incorporating conservation errors into the loss function, as named physical informed neural networks (PINNs) [35]. Eq. (53) shows the definition of the unscaled residual for the pressurebased solver for the continuity equation in general commercial CFD codes [36]. The rate of mass creation in each cell was calculated as the sum of the mass change in the control volume inflow/outflow amount into the control volume. Flow variables represent the right, left, top and bottom grid surfaces.

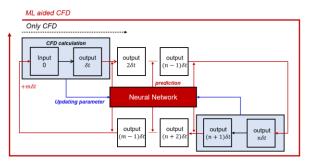


Figure 13. Concept of machine learning aided CFD network with sequential parameter update via residual monitoring.

$$R_{t}^{\phi} = \sum_{grids\,n} \left| \left(\frac{\rho_{m}^{t} - \rho_{m}^{t-1}}{\Delta t} \right) - \left(\frac{\rho_{v}^{t} v_{r,r}^{t} - \rho_{l}^{t} v_{r,l}^{t}}{\Delta x} + \frac{\rho_{v}^{t} v_{z,t}^{t} - \rho_{b}^{t} v_{z,b}^{t}}{\Delta r} \right) + \frac{\rho_{m}^{t} v_{r,m}^{t}}{r} \right| \tag{53}$$

In this study, the residual was scaled through the residual values in the training dataset which matched the tolerance level of CFD simulation (Eq. (54)). In CFD simulations, the residual was scaled by the dominator which is the largest absolute value in the first five iteration. The navy line in Fig. 10

shows the calculated the continuity residual in the multi-step prediction case. It should be noted that the residual increase slope was gentle although the next time series data was calculated only by the neural network model without the first principles. It seems that the FVMN model effectively embody the conservativeness of the FVM. However, since the discrepancies of input variables between the training dataset and prediction datasets became deepens, the mass conservation error in the trained networks gradually increased.

$$R_t^c = \frac{R_t^{\phi}}{R_{0.601}^{\phi}} \tag{54}$$

As a result, we preliminarily confirmed that the degradation of the neural networks can be estimated by the residual calculation without the ground truth flow fields. Although only the continuity residual was observed in this study, this methodology can be reliable with simultaneous calculations for the momentum and energy conservation. The adaptation of the PINNs algorithm, which includes the conservation errors in the loss function, can also improve the MACnet feasibility. As discussed in Section 1, the recent benchmark studies show that about 10 – 100 h CPU time per 1 s physical time is required to simulate unsteady hydrogen deflagration in the compartment unit [4]. If the suggested MACnet framework is more concreted with various case studies, it will be a promising solution to overcome the limitation of complex reacting flow simulation.

5. CONCLUSION

In this study, a new concept of network model was proposed to more accurately predict the multi-step CFD time series data. The developed FVMN can sufficiently consider the finite volume method. which is the basic principle of most CFD codes, by preprocessing process of input/output variables. We evaluated the individual efficacy of the tier-input and derivative-output system based on an unsteady reacting flow simulation. We confirmed that the scale separation of output variables by the derivative system has a significant effect on improving networks performance. In addition, as we expected, the tier system can improve the networks performance of calculating the transport of physical quantities between grids. Although the relative error with the ground truth data was significantly reduced by the developed network model, the linearly increasing gradient error is a remaining issue in transient calculations for longer

times. Therefore, we suggested Machine learning aided CFD framework which can accelerate the CFD simulation through alternating computations. We preliminarily confirmed that the interval of the cross calculation can be determined by the residual calculation without the ground truth data. Since this study aims to develop a baseline model, improvement of the model such as data sampling and physics informed loss function is our future work.

ACKNOWLEDGEMENT

This work was supported by the Nuclear Safety Research Pro- gram through the Korea Foundation of Nuclear Safety (KoFONS) using the financial resource granted by the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea (grant number 20 030 06-0120-CG10 0).

REFERENCES

- [1] A. Lipatnikov, T. Nilsson, R. Yu, X. Bai, V. Sabelnikov, Assessment of a flamelet approach to evaluating mean species mass fractions in moderately and highly turbulent premixed flames, *Physics of Fluids* 33(4) (2021) 045121.
- [2] S. Lakshmipathy, T. Skjold, H. Hisken, G. Atanga, Consequence models for vented hydrogen deflagrations: CFD vs. engineering models, *International Journal of Hydrogen Energy* 44(17) (2019) 8699-8710.
- [3] X. Wen, L. Dressler, A. Dreizler, A. Sadiki, J. Janicka, C. Hasse, "Flamelet LES of turbulent premixed/stratified flames with H2 addition," Combustion and Flame 230 (2021) 111428.
- [4] I. Tolias et al., Numerical simulations of vented hydrogen deflagration in a medium-scale enclosure, *Journal of Loss Prevention in the Process Industries* 52 (2018)125-139.
- [5] I. Tolias et al., Best practice guidelines in numerical simulations and CFD benchmarking for hydrogen safety applications, *International Journal of Hydrogen Energy* 44(17) (2019) 9050-9062.
- [6] C. Metrow, S. Gray, G. Ciccarelli, Detonation propagation through a nonuniform layer of hydrogen-oxygen in a narrow channel, *International Journal of Hydrogen Energy* (2021) In press.
- [7] J. Jeon, D. Shin, W. Choi, S. J. Kim, Identification of the extinction mechanism of lean limit hydrogen flames based on Lewis number effect, *International Journal of Heat and Mass Transfer* 174 (2021) 121288
- [8] N. K. Kim, J. Jeon, W. Choi, S. J. Kim, Systematic hydrogen risk analysis of OPR1000 containment before RPV failure under station blackout scenario, *Annals of Nuclear Energy* 116 (2018) 429-438.
- [9] R. T. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, arXiv preprint arXiv:1806.07366, 2018.

- [10] S. Kim, W. Ji, S. Deng, C. Rackauckas, Stiff Neural Ordinary Differential Equations, *arXiv* preprint *arXiv*:2103.15341, 2021.
- [11] S. Ledesma, D.-L. Almanza-Ojeda, M.-A. Ibarra-Manzano, E. C. Yepez, J. G. Avina-Cervantes, P. Fallavollita, Differential Neural Networks (DNN), *IEEE Access* 8(2020) 156530-156538.
- [12] A. Takbiri-Borujeni, M. Ayoobi, Application of physics-based machine learning in combustion modeling, 11th US National Combustion Meeting, California, USA, March, 2019.
- [13] S. Lee, D. You, Data-driven prediction of unsteady flow over a circular cylinder using deep learning, *Journal of Fluid Mechanics* 879 (2019)217-254.
- [14] T. Praditia, M. Karlbauer, S. Otte, S. Oladyshkin, M. V. Butz, W. Nowak, Finite Volume Neural Network: Modeling Subsurface Contaminant Transport, *arXiv* preprint arXiv:2104.06010, 2021.
- [15] N. Amangeldiuly, D. Karlov, M. V. Fedorov, Baseline Model for Predicting Protein–Ligand Unbinding Kinetics through Machine Learning, Journal of Chemical Information and Modeling 60(12) (2020) 5946-5956.
- [16] P. A. Whigham, C. A. Owen, S. G. Macdonell, A baseline model for software effort estimation, *ACM Transactions on Software Engineering and Methodology (TOSEM)* 24(3) (2015) 1-11.
- [17] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, *arXiv preprint arXiv:1611.06455*, 2016.
- [18] J. Guerrero, A Crash Introduction to the Finite Volume Method and Discretization Schemes in OpenFOAM, 15th OpenFOAM workshop, VA, USA, June, 2020.
- [19] F. Moukalled, L. Mangani, and M. Darwish, The finite volume method in computational fluid dynamics, Springer, 2016.
- [20] J. A. Anderson, An introduction to neural networks, MIT press, 1995.
- [21] V. Sharma, S. Rai, A. Dev, A comprehensive study of artificial neural networks, *International Journal of Advanced research in computer science and software engineering* 2(10) (2012).
- [22] S. Hayou, A. Doucet, J. Rousseau, On the impact of the activation function on deep neural networks training, *arXiv* preprint arXiv:1902.06853, 2019.
- [23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, *The journal of machine learning research*, 15 (1) (2014) 1929-1958.
- [24] D. Svozil, V. Kvasnicka, J. Pospichal, Introduction to multi-layer feed-forward neural networks, *Chemometrics and intelligent laboratory systems*, 39(1) (1997) 43-62.
- [25] J. Ren, L. Xu, On vectorization of deep convolutional neural networks for vision tasks, *Proceedings of the AAAI Conference on Artificial Intelligence* 29 (1) (2015).
- [26] M. Buscema, Back propagation neural networks, Substance use & misuse 33(2) (1998) 233-270.
- [27] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, *nature 323*(6088) (1986) 533-536.
- [28] A. Ansari, S. Mohaghegh, M. Shahnam, J. Dietiker, A. Takbiri Borujeni, E. Fathi, Data Driven Smart Proxy for CFD: Application of Big Data Analytics &

- Machine Learning in Computational Fluid Dynamics, Part One: Proof of Concept National Energy Technology Lab.(NETL), Pittsburgh, PA, 2017.
- [29] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.
- [30] J. Snoek, H. Larochelle, R. P. Adams, Practical bayesian optimization of machine learning algorithms, *arXiv preprint arXiv:1206.2944*, 2012.
- [31] Y. Shoshin, L. Tecce, J. Jarosinski, Experimental and computational study of lean limit methane-air flame propagating upward in a 24 mm diameter tube, *Combustion Science and Technology* 180(10-11) (2008) 1812-1828.
- [32] F. E. Hernández-Pérez, B. Oostenrijk, Y. Shoshin, J. A. van Oijen, L. P. de Goey, Formation, prediction and analysis of stationary and stable ball-like flames at ultra-lean and normal-gravity conditions, Combustion and Flame 162(4) (2015) 932-943.
- [33] J. Jeon, H. Jung, Y.S. Kim, S.J. Kim, Numerical study of lean limit hydrogen flames propagating upward to validate a flammability limit model, 18th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (2019) 4362-4371.
- [34] M. Rahman, I. Pop, M. Saghir, Steady free convection flow within a titled nanofluid saturated porous cavity in the presence of a sloping magnetic field energized by an exothermic chemical reaction administered by Arrhenius kinetics, *International Journal of Heat and Mass Transfer* 129 (2019) 198-211.
- [35] K. Shukla, P. C. Di Leoni, J. Blackshire, D. Sparkman, and G. E. Karniadakis, Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks, *Journal of Nondestructive Evaluation* 39(3) (2020) 1-20.
- [36] ANSYS FLUENT 18.0 Theroey Guide, 2017.

[Supplementary materials]

Table S1. Test matrix for optimization of FVMN

Case	Hidden layers	Number of parameters	Activation function	Learning rate	Loss function
a	64	2,049	ReLU	0.001	MSE
b	64, 64	6,209	ReLU	0.001	MSE
c	64, 64, 64	10,369	ReLU	0.001	MSE
d	64, 64, 64, 64	14,529	ReLU	0.001	MSE
e	64, 64, 64	10,369	Sigmoid	0.001	MSE
f	128, 128, 128	37,121	ReLU	0.001	MSE
g	256, 256, 256	139,777	ReLU	0.001	MSE
h	64, 32, 16	4,609	ReLU	0.001	MSE

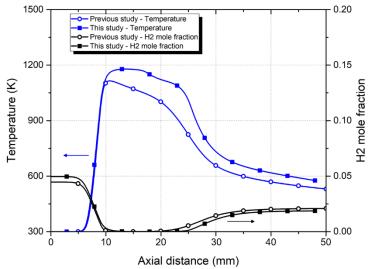


Figure S1. Verification of the transient simulation results through comparison with the previous steady results. Since the initial hydrogen concentration was as high as 0.5% in this study, the maximum temperature also increased slightly.

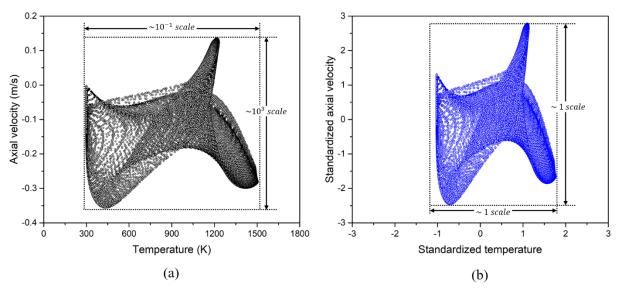


Figure S2. Before (a) and after (b) standardization of features to resolve large differences between their ranges. It was confirmed that the features including temperature and axical velocity were transformed to comparable scales.

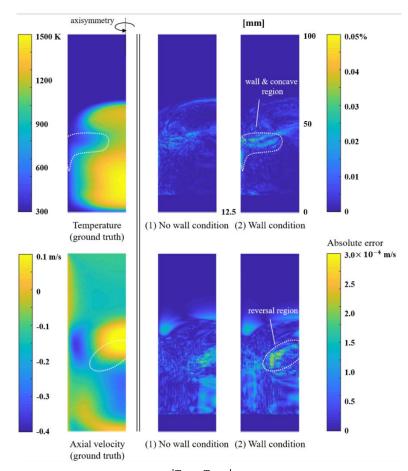


Figure S3. Error fields of predicted temperature $\frac{|T_{ML}-T_{CFD}|}{T_{CFD}}$ and axial velocity $|u_{ML}-u_{CFD}|$ according to the presence of the wall condition. The error in the gradient region of each variable became more pronounced at the wall condition.

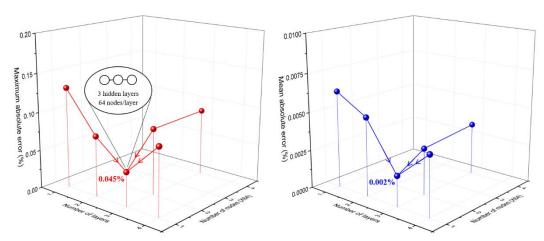


Figure S4. Changes in maximum absolute error (left) and mean absolute error (right) according to the number of hidden layers and nodes.