Design Support Techniques for Turbocharger using Machine Learning and CAE

SAITO Hiroki : Computational & Mathematical Engineering Group, Technology Platform Center, Technology & Intelligence Integration
HATTORI Hitoshi : Computational & Mathematical Engineering Group, Technology Platform Center, Technology & Intelligence Integration
YONEKURA Kazuo : Doctor of Engineering, Senior Researcher, Computational & Mathematical Engineering Group, Technology Platform Center, Technology & Intelligence Integration (Doctor of Engineering, Lecturer, Department of Systems Innovation, Graduate School of Engineering, The University of Tokyo)

Recently CAE is becoming widely used in the design of turbomachinery, such as turbochargers and aircraft engines, to predict its performance. In some specific cases, however, it is difficult to predict the performance using CAE. This is mainly because the physical phenomena are not completely clarified due to their complexities. Furthermore, CAE is also used in combination with the optimization techniques, such as genetic algorithms and response surface methods, to obtain a better design solution. Although many simulations are performed and accumulated, these optimization techniques cannot utilize the accumulated knowledge. In this research, two machine learning techniques, deep neural network and deep Q-network, are applied to two problems respectively: performance prediction of the volumetric flow at the surge of compressors and shape optimization for minimizing pressure drop of the low-pressure turbine airfoils. The results show that deep neural network has high predictive accuracy while not requiring physical models. It has also been confirmed that deep Q-network acquires high generalization capabilities to be applied to different scenarios by training with various conditions. In addition, it has been revealed that deep Q-network puts large weight at the same point as well-trained designers do.

1. Introduction

In recent years, CAE (Computer Aided Engineering) has become widely used in the design of turbomachinery, such as turbochargers and aircraft engines. Using CAE, a product can be modeled to the extent within which analysis is possible, and its performance can be predicted through numerical analysis. In some specific cases, however, it is not possible to adequately predict performance with CAE only, and a typical example of this is compressor surge.

In a vehicle turbocharger, if a self-excited vibration phenomenon accompanied by large amplitude pressure and flow rate fluctuations, known as surge, occurs in the whole piping system, including the compressor, then not only does the turbocharger fail to fulfill its role, but the pressure fluctuation and vibrations may damage the compressor and its peripheral equipment. Therefore, surge is considered as a phenomenon to be avoided at any cost, and when expanding the operating range of compressors, it is important to predict the volumetric flow at the peak pressure ratio point (volumetric flow rate at surge) in the design phase. A common method of predicting the volumetric flow rate at surge is to predict overall compressor performance, i.e., pressure ratio-flow characteristics, by performing a three-dimensional steady flow analysis with CFD (Computational Fluid Dynamics) at multiple operating points. However, at the preliminary design phase, the three-dimensional shape has not yet been determined, and it is unrealistic in terms of calculation cost to conduct a CFD analysis for every possible shape. Even if the three-dimensional shape has been determined, at the low flow rates close to the volumetric flow rate at surge, performance could deteriorate due to unsteady phenomena such as rotating stall, and the calculation tends to be unstable. Therefore, using steady flow analysis, it is often difficult to accurately predict overall compressor performance. Even if unsteady flow analysis is performed, it is difficult to quantitatively reproduce the volumetric flow rate at surge and sufficiently understand the flow field at the low flow rates close to it. For these reasons, there is no established technique for predicting the volumetric flow rate at surge using CFD in the design phase and, at present, it is qualitatively estimated based on past data for similar shapes, etc., which depends on the knowledge of the designers. For this reason, reconsidering design methodology tends to be required, resulting in increased product development cost. In order to establish a prediction technique, it is important to elucidate the surge phenomenon and develop physical models, but a large amount of research is still currently required. In such a situation, it is considered effective to develop a system that uses physical models to predict the known part of the phenomenon, and machine learning based on data to predict the unknown part. The reliability of such a system will be enhanced by replacing the machine learning part with physical models as understanding of the

phenomenon progresses.

In addition, the purpose of using CAE is to obtain a better design solution, and searching for this design solution involves repeated performance of many numerical analyses. These many trials and analytical results constitute a precious asset, and a better design solution can be obtained more quickly by analyzing and using them effectively. Hitherto, the evolutionary algorithm[1] and the response surface methodology[2] have been used for optimization, but with these techniques, the analysis must be redone from the beginning if the flow boundary conditions are different and past analysis results cannot be used, so that use of past analysis results as an asset is difficult. However, for example, even with new design conditions, experienced designers can make deductions based on their own design experience and, in a similar manner, there should be some method of using the knowledge obtained from past analyses even with different flow boundary conditions. Research is underway to achieve such capabilities through machine learning[3], and these are known as generalizability because they can be used not only under specific conditions but also for general purposes. This paper discusses reinforcement learning, which is a relatively new machine learning technique characterized by a high level of generalizability. If this high generalizability can be used in the design phase, it may be possible to reduce analysis time to less than that required by conventional optimization techniques. In addition, feeding back the design points learned by machine learning to the designers may enable them to verify the soundness of the machine learning model and obtain new findings. This paper also describes the potential of such feedback.

As mentioned above, this paper describes a prediction technique for volumetric flow rate at surge which uses a DNN (Deep Neural Network)[4] to complement the unknown part of the physical model, and describes a design solution search technique which uses the high generalizability of a DQN (Deep Q-Network)[5] — a reinforcement learning technique — together with past analysis data.

2. Design support techniques using machine learning and CAE

2.1 Overview of machine learning techniques

2.1.1 Deep Neural Networks

In recent years, machine learning has been attracting attention as a technique for discovering data features and the laws existing between different groups of data, and for predicting unknown values, with NN (Neural Networks) being a representative machine learning technique. Using NN, a complicated nonlinear function can be represented by connecting multiple nodes in layers and changing the connection strength between nodes, called the weight, through training. Even for a complicated phenomenon such as a surge, it is considered possible to predict the volumetric flow rate at surge by associating the shape parameters and volumetric flow rate at surge based on abundant data. Examples of application of NN to mechanical products include prediction and optimization of engine efficiency and pollutant emissions such as NO_x[6, 7], prediction of misfire in diesel engines[8], and prediction of the position of the reluctance motor rotor[9]. In addition, there have been reports of NN being used to predict the compressive strength of concrete based on the amounts of slag and fly ash[10].

Figure 1 shows the structure of a four-layered NN, which has two intermediate layers between the input layer and output layer. The NN in **Fig. 1**, in which all nodes are connected across different layers, is called a FCNN (Fully-Connected Neural Network). The NN consists of input nodes, bias nodes, summation nodes, and activation function nodes. Input nodes pass the received value directly to the next layer; bias nodes always pass 1; summation nodes pass the sum of the received values to the next layer; and activation function nodes pass the received value to the next layer after processing it with a function. The summation and activation function nodes perform the following calculations and pass the output to the next layer.

$$\begin{aligned} o^l_{i,\mathrm{sum}} &= b^l_i + w^l_{i1} o^{l-1}_{1,\ \mathrm{func}} + w^l_{i2} o^{l-1}_{2,\ \mathrm{func}} + \cdots \\ &\quad + w^l_{iN^{l-1}} o^{l-1}_{N^{l-1},\ \mathrm{func}} \\ &= b^l_i + \sum_{k=1}^{N^{l-1}} w^l_{ik} o^{l-1}_{k,\ \mathrm{func}} \qquad \cdots\cdots(1) \end{aligned}$$

$$o^l_{i,\mathrm{func}} = f\left(o^l_{i,\mathrm{sum}}\right) \qquad \cdots\cdots(2)$$

In these equations, $o^l_{i,\mathrm{sum}}$ and $o^l_{i,\mathrm{func}}$ represent the outputs of the i-th summation node and activation function nodes of the l-th layer where the input layer is $l = 0$, b^l_i represents bias, w^l_{ik} represents the weight of the connection from the k-th node of the $(l-1)$-th layer to the i-th node of the l-th layer, and N^{l-1} represents the number of nodes in the $(l-1)$-th layer.

$f(\cdot)$ is called an activation function, for which a nonlinear function, such as Sigmoid, tanh, ReLU[11], or leaky ReLU[12] is commonly used. The final output is obtained by calculating Equations (1) and (2) repeatedly for all nodes from the input layer to output layer, so that, generally, an NN with a larger number of layers and nodes can represent a more complicated function. An NN having two or more intermediate layers is called a DNN.

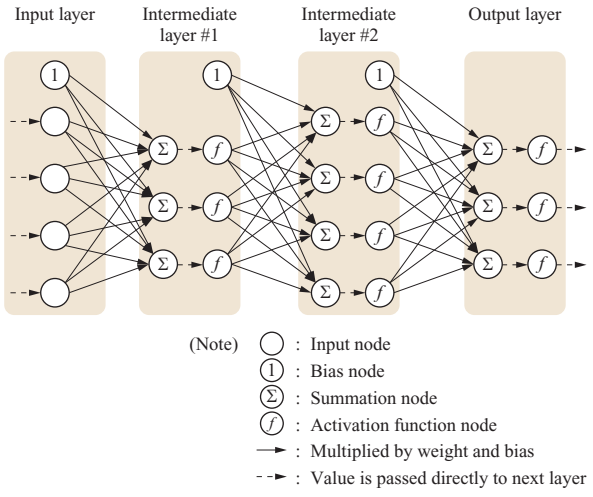


Fig. 1 Architecture of NN

One type of NN training is called supervised training, in which input x and corresponding answer t are given, and weight w and bias b are updated so that the output y of the NN is as close to answer t as possible. The tasks performed in supervised training are regression and classification, and, in this paper, regression for supervised training is performed in order to predict volumetric flow rate at surge based on past test data. To update the weight for regression, the following equation is used:

$$w_{ik}^{l} \leftarrow w_{ik}^{l} - \eta \frac{\partial E}{\partial w_{ik}^{l}}. \quad \cdots\cdots(3)$$

In this equation, η is the learning coefficient, and E is the error function. The following square error is commonly used in regression:

$$E = \frac{1}{2}\sum_{k=1}^{N^L}\left(t_k - o_{k,\,\mathrm{func}}^{L}\right)^2 + \frac{1}{2}\lambda\sum_{l=1}^{L}\sum_{i=1}^{N^l}\sum_{k=1}^{N^{l-1}}\left(w_{ik}^{l}\right)^2. \quad \cdots\cdots(4)$$

N^L is the number of nodes in the output layer, $o_{k,\,\mathrm{func}}^{L}$ is the output of the output layer (= y_k), and t_k is the corresponding answer. The second term is a normalization term for suppressing overtraining, which is described below, and λ is the normalization rate, which represents the degree of suppression. Training coefficient η in Equation (3) is related to the training speed and whether or not the solution is local, and various algorithms have been proposed for optimizing the training coefficient through training. Among such algorithms, Adam[13] is commonly used as one that provides stable, accurate training results.

In supervised training by machine learning, including NN, overtraining sometimes occurs, which is a phenomenon in which overfitting to the training data results in poor prediction performance with unknown data. Normalization in the second term of Equation (4) is introduced to suppress overtraining, but dropout[14] is also commonly used. Dropout is a technique that excludes a certain percentage of the nodes in the intermediate layers from training. Each time training data are given to the NN, the nodes to be excluded change randomly. This makes it possible to obtain the same results as when the results of training with multiple NNs are averaged, which works as an effective means of suppressing overtraining.

2.1.2 Deep Q-Network

RL (Reinforcement Learning) is a machine learning technique which involves learning through trial and error the action that will maximize a value which is set in a specific environment. Examples of use of RL include AlphaGo[15], which won a game of Go against a professional player, robot arms[16], and autonomous driving of automobiles[17].

Figure 2 shows the basic structure of RL, which consists of an environment and an agent that determines what action to take in response to that environment.

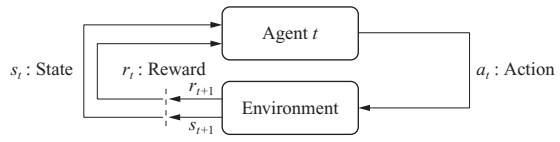


Fig. 2 Schematic image of RL[18]

The agent selects the next action to take a_t based on state s_t at the t-th attempt, and sends it to the environment. The environment sends the state s_{t+1}, which has changed in response to the received action a_t, and the previously set reward r_{t+1} for that action to the agent. Various RL techniques have been proposed that adopt different ways of selecting the action according to the state; in one of these, Q learning, the value of the action is determined using the Q function (action value function), which can be represented by the following equation, and the agent selects the action with the highest value.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left\{r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)\right\} \quad \cdots\cdots(5)$$

In this equation, α ($0 < \alpha \leq 1$) is the learning rate, γ ($0 < \gamma \leq 1$) is the discount factor, and A represents all the actions that can be taken. The second term of Equation (5) represents the difference between the expected and current values of the action value, and the current action value is updated by this difference. In actual Q learning, the Q function is created using table functions, and it is necessary to hold values corresponding to every possible state and combination of actions. However, if the state is obtained as an image, such as in AlphaGo and autonomous driving, the amount of Q function data is enormous, making training difficult. To overcome this problem, DQN uses an NN as an approximation function for the Q function.

DQN approximates the Q function using supervised training, but no answer can be obtained because it is not possible to know the true value of the Q function. Therefore, the expected value for the action value in Equation (5) is used as the answer. In this case, the error function E_{DQN} for the NN is as follows:

$$E_{\mathrm{DQN}} = \frac{1}{2}\left\{r_t + \gamma \max_{a \in A} Q_\theta(s_{t+1}, a) - Q_\theta(s_t, a_t)\right\}^2. \quad \cdots\cdots(6)$$

In this equation, Q_θ represents the approximate Q function obtained by using the NN. A locally-optimized Q function can be obtained by performing training such that the second term on the right side of Equation (5), which corresponds to the update amount of the Q function, is zero.

As previously described, if the state is obtained as an image, then a CNN (Convolutional Neural Network)[19], shown in **Fig. 3**, is used. A CNN consists of convolutional layers, which extract feature values, and pooling layers, which decrease the number of dimensions. In the convolutional layers, the same operation as the filtering used in image processing is performed using the weights of the network, and the filter is updated through training, thereby enabling extraction of important feature values. In the pooling layers, the mean and maximum values in a certain range are extracted, but there is no updating through training. In image recognition by CNN, the output is generally obtained by using an FCNN after using a combination of convolutional and pooling layers multiple times. When a CNN is used in the DQN, the next action is determined based on the action

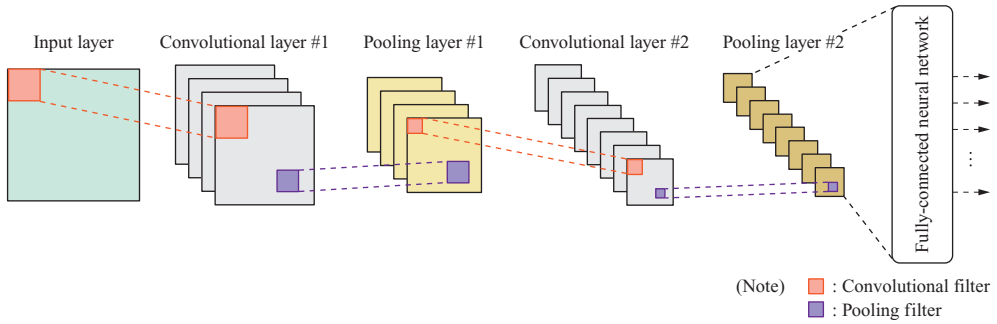


Fig. 3 Architecture of CNN

value obtained from the image. This mechanism imitates the process by which humans learn and select their actions based on visual information.

2.2 Prediction of volumetric flow rate at surge using DNN

In this study, we used a DNN to predict the volumetric flow rate at surge of a turbocharger, and constructed a prediction model using past test data of actual products.

2.2.1 Data used

We used pressure ratio-flow rate characteristics data measured in the past, as shown in **Fig. 4**. Taking the volumetric flow rate at surge to be the volumetric flow rate at peak pressure ratio on a third-order spline curve fitted to the measurement points, we obtained the volumetric flow rate and pressure ratio at surge, and used a DNN to create a regression model that predicts the surge flow rate coefficient ϕ_{surge} and pressure coefficient μ_{surge}, which are calculated using Equations (7) and (8).

$$Q_{surge} = \phi_{surge} N_{rot} \pi D^3 \quad \cdots\cdots(7)$$

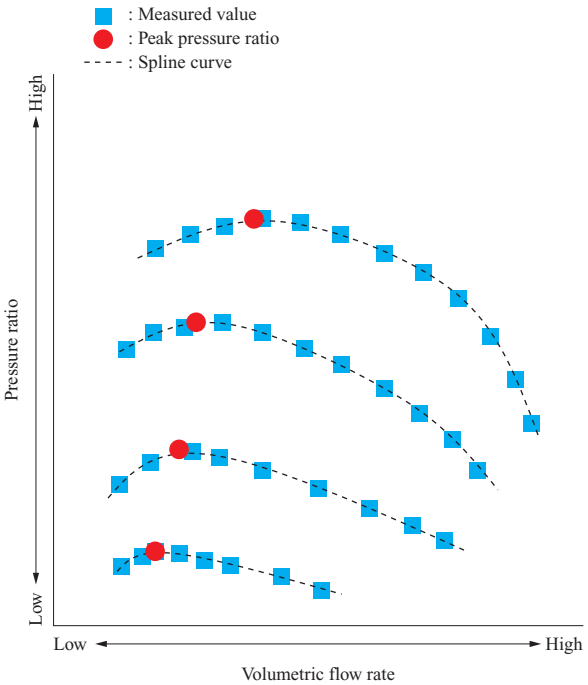


Fig. 4 Characteristics between pressure ratio and volumetric flow rate and peak value of pressure ratio

$$\Pi_{surge} = \left\{ \mu_{surge} \left(Mu\sqrt{T_0 \gamma R} \right)^2 \left(\frac{\gamma - 1}{T_0 \gamma R} + 1 \right) \right\}^{\frac{\gamma}{\gamma - 1}} \quad \cdots\cdots(8)$$

In these equations, Q_{surge} and Π_{surge} are the volumetric flow rate and pressure ratio at surge, respectively, N_{rot} is the rotational speed, D is the outer diameter of the impeller, and Mu is the impeller tip Mach number. T_0 is 293.15 K, γ is 1.4, and R is 287 J/kg·K. There were data for 102 product types, and volumetric flow rate and pressure ratio at surge were obtained for four to seven different impeller tip Mach number per product type, so that the total number of samples was 574. As input parameters for training, we used a total of 104 parameters consisting of 100 shape parameters that were selected by experts with abundant knowledge of turbochargers as being related to the occurrence of surge, 1 impeller tip Mach number, and 3 predicted values for surge flow rate coefficient obtained from 3 simple, one-dimensional physical models.

In not only DNNs but also other types of machine learning, data division techniques are used to evaluate the training results. Holdout is a data division technique in which data are divided into training and verification data, and prediction accuracy is verified with the verification data after training with the training data. Since the verification data are not used for training, the prediction accuracy for unknown data can be evaluated. Cross validation is another data division technique, in which data are divided into several groups, one of which is used to verify the prediction accuracy after training is performed with the other groups[20]. Exchanging the group used for verification each time, this is performed the same number of times as the number of groups, and the prediction accuracies obtained are averaged to give the final prediction accuracy. Since cross validation allows all the prepared data to be used for training, it is often used to set the parameters used in training. In this study, holdout was used. Of 102 product types, 6 (number of samples: 31) were used as verification data, and the remaining 96 (number of samples: 543) as training data.

2.2.2 Hyper parameters for DNN

A DNN involves a large number of parameters — such as number of nodes and normalization rate — which are called hyper parameters. In addition to these being determined based on the user's experience, use is also made of the grid search, which tries all combinations of hyper parameters, random search, which changes hyper parameter values

randomly within a specified range, and Bayesian optimization[21], which is a type of optimization technique. In this study, Bayesian optimization was used to determine hyper parameters. Using Bayesian optimization, the hyper parameters that are expected to minimize the prediction error can be obtained by treating the DNN as a prediction error function that has the hyper parameters as its variables, and approximating the prediction error for the hyper parameters using a probabilistic regression technique, such as Gaussian process regression.

When determining hyper parameters by Bayesian optimization, the above-mentioned cross validation — with the number of divisions set to 5 — was used to evaluate the square error of the predicted values for volumetric flow rate at surge. Chainer ver. 3.1.0[22] was used for DNN training, and GPyOpt ver. 1.0.3[23] for Bayesian optimization. Five hyper parameters were optimized: number of epochs, batch size, number of nodes in an intermediate layer, number of intermediate layers, and dropout rate. All the intermediate layers had the same number of nodes. The other hyper parameters were given fixed values, the activation function was leaky ReLU, the optimization technique for the training coefficient was Adam, the normalization rate was λ = 0.000 5, and the initial value distribution for weight was LeCun Normal (mean 0, standard deviation $1/\sqrt{n}$ (n: number of nodes in each layer)). **Table 1** shows the search ranges for the hyper parameters and optimal values obtained by Bayesian optimization. This table shows the optimized results for training for volumetric flow rate at surge, but the same values were used for training for pressure ratio at surge.

2.2.3 Results and discussion

We evaluated the results obtained in this section by converting the predicted surge flow rate coefficient ϕ_{surge} and pressure coefficient μ_{surge} to volumetric flow rate and pressure ratio at surge using Equations (7) and (8).

We first verified the effectiveness of the DNN by comparing its training results to those of three other machine learning techniques: SVM (Support Vector Machine), RF (Random Forest), and KRR (Kernel Ridge Regression). For those three techniques, we determined hyper parameters by random search and used Scikit-learn ver. 0.19.1[24] for training. **Table 2** shows the RMSE (Root Mean Squared Errors) for the predicted volumetric flow rate at surge for the six product types used as verification data. The RMSE is defined by the following equation:

$$\mathrm{RMSE} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(t_i - y_i\right)^2}\,. \quad \cdots\cdots(9)$$

Table 1 Hyper parameters optimized by Bayesian optimization

Hyper parameter	Unit	Search range	Optimal value
Number of epochs	epochs	1 000 to 3 000	2 204
Batch size	items/batch	100 to 600	543
Number of nodes in an intermediate layer	nodes	50 to 520	450
Number of intermediate layers	layers	3 to 10	6
Dropout rate	—	0.000 to 0.510	0.316

Table 2 Performance comparison of DNN and other machine learning methods

Product type	RMSE for volumetric flow rate at surge (kg/m^3)			
	DNN	SVM	RF	KRR
A	0.229 7	0.242 1	0.468 3	0.742 3
B	0.032 4	0.807 7	0.188 5	0.476 8
C	0.083 7	0.276 6	0.316 1	0.663 0
D	0.101 3	0.473 3	0.141 2	0.302 2
E	0.196 8	0.319 5	0.176 5	0.162 9
F	0.387 4	0.442 8	0.352 7	0.447 2
Whole of verification data	0.208 3	0.448 0	0.293 3	0.494 1

In this equation, M is the number of samples, and t_i and y_i are the answer and predicted value for sample i, respectively. Looking at the results for the whole of the verification data in **Table 2**, it can be seen that the DNN has higher prediction accuracy than the other machine learning techniques. Even when individual product types are compared, the DNN has the highest accuracy in most cases, and even where another technique is more accurate, the difference is very small. From the above, it was confirmed that the DNN is effective for predicting volumetric flow rate at surge. Since the surge phenomenon is highly nonlinear, it may be presumed that the DNN, which can represent nonlinearity better than the other techniques, is more effective.

Figure 5 shows the pressure ratio-flow rate characteristics, and the measured and DNN-predicted volumetric flow rates and pressure ratios at surge for the six product types used as verification data. On both the vertical and horizontal axes, the predicted values are the values predicted by the DNN. The mechanism of surge occurrence differs depending on the operating conditions of the compressor, and this is thought to affect prediction accuracy. **Table 3** shows the volumetric flow rate at surge for the six product types used as verification data, the RMSE of the pressure ratio at surge for each product type, and the RMSE for the low-speed (Mu < 1.3) and high-speed (Mu > 1.3) conditions. Focusing on the volumetric flow rate at surge in **Table 3**, it can be seen that product type F has lower prediction accuracy than any other product type. **Figure 6** shows histograms, created from the training data, of the specific shape parameters α and β which are considered to strongly affect the surge phenomenon, and also shows to which bin of the histogram the shape parameters α and β for product types A to F belong. From **Fig. 6**, it can be seen that the training data contain a relatively large number of shape parameter α and β values at the same level as for product types A to E, but that there are few or no such values for product type F, and it is presumed that prediction accuracy was affected by the difference in the relative amounts of data contained in the training data.

In **Table 3**, focusing on the differences between the low-speed and high-speed conditions for volumetric flow rate and pressure ratio at surge, it can be seen that prediction accuracy tends to be low in the high-speed condition regardless of the product type. In the training data, the ratios

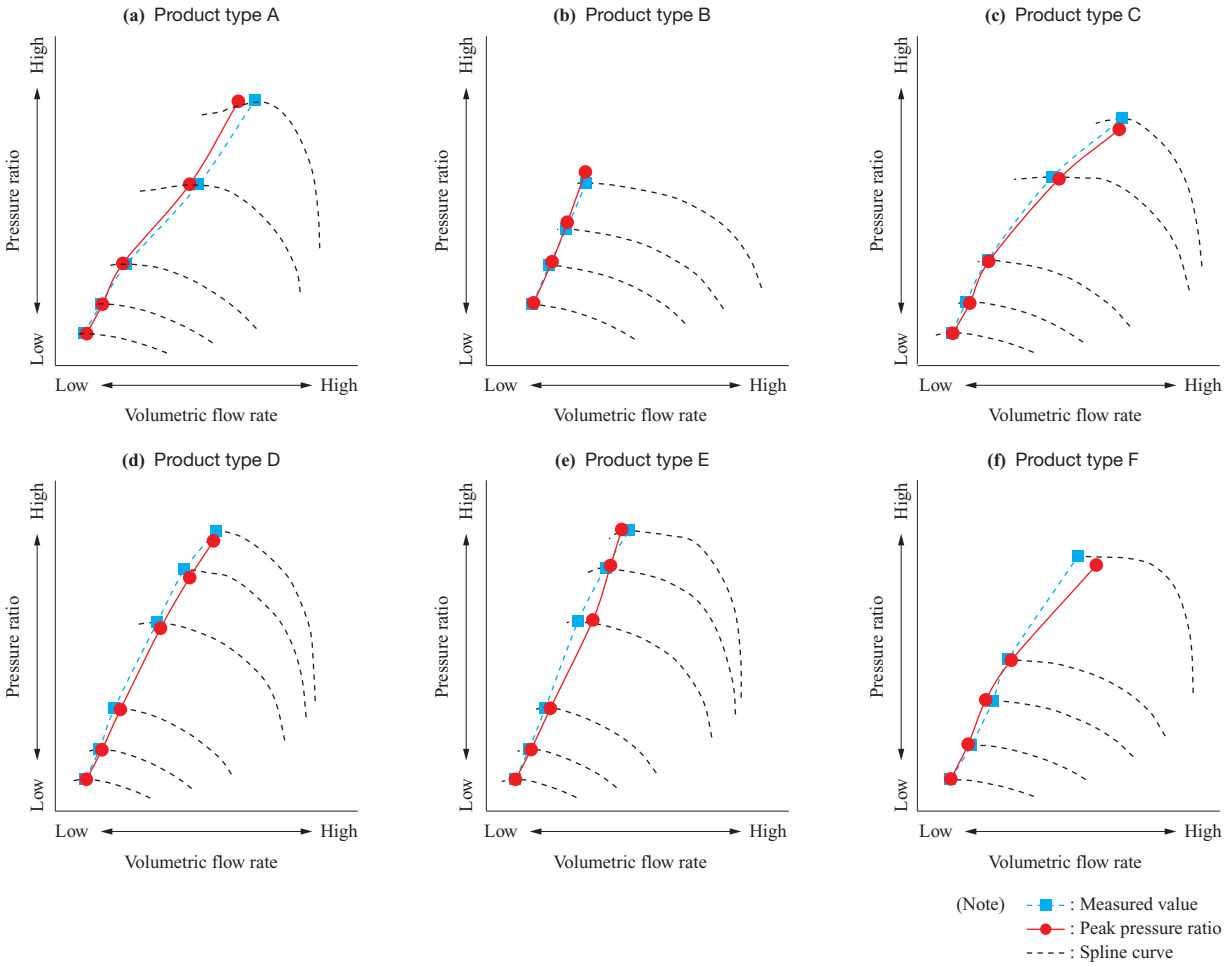


Fig. 5 Prediction of volumetric flow rate at surge and pressure ratio at surge for test data

Table 3 RMSE of prediction of volumetric flow rate and pressure ratio at surge for test data

Product type	RMSE for volumetric flow rate at surge (kg/m^3)			RMSE for pressure ratio at surge (–)		
	Total	$Mu < 1.3$	$Mu > 1.3$	Total	$Mu < 1.3$	$Mu > 1.3$
A	0.229 7	0.076 0	0.349 4	0.005 3	0.004 2	0.006 5
B	0.032 4	0.036 5	0.014 7	0.060 9	0.040 3	0.099 8
C	0.083 7	0.048 5	0.118 2	0.046 2	0.012 2	0.071 5
D	0.101 3	0.098 2	0.104 2	0.052 9	0.008 8	0.074 2
E	0.196 8	0.081 9	0.266 0	0.010 5	0.004 6	0.014 2
F	0.387 4	0.180 5	0.787 5	0.036 8	0.006 1	0.081 3
Whole of verification data	0.208 3	0.104 6	0.307 9	0.040 1	0.017 5	0.060 6

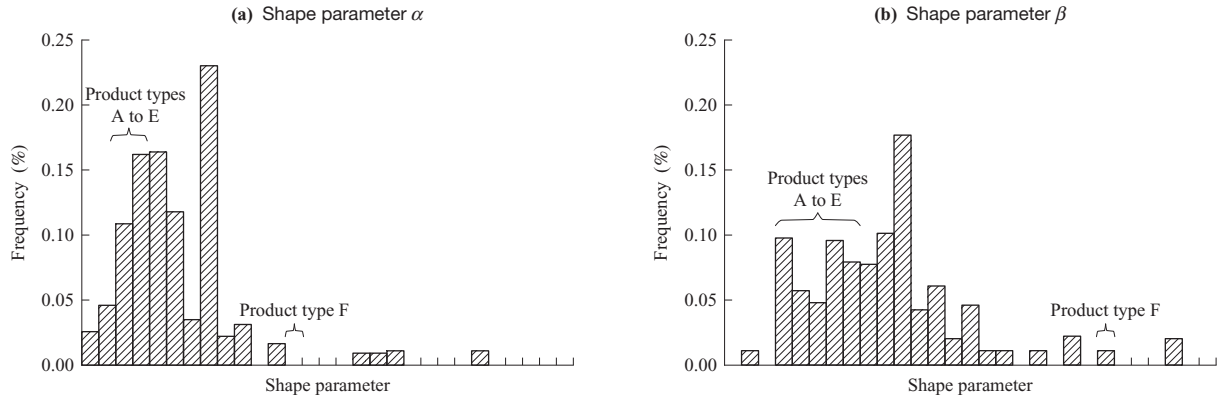


Fig. 6 Histograms of shape parameters having an effect on surge

of the number of samples for the low-speed and high-speed conditions are respectively 62% (339 samples) and 38% (204 samples) of the total number of samples, and so one cause of the differences in prediction accuracy is believed to be the relatively small number of samples for the high-speed condition. The number of samples was different because measurement in the high-speed condition is more difficult than in the low-speed condition.

The RMSE is related to the loss function of the DNN, and is therefore suitable for evaluating the process of convergence but not for intuitively evaluating the prediction error. Therefore, instead of RMSE, we used the MRE (Mean Relative Error), which is a relative error defined in relation to the true value, to evaluate prediction performance. MRE is defined by the following equation:

$$\mathrm{MRE} = \frac{1}{M}\sum_{i=1}^{M}\left|\frac{t_i - y_i}{t_i}\right|. \quad \cdots\cdots(10)$$

From the MRE values shown for the whole of the verification data in **Table 4**, it can be seen that the MRE for volumetric flow rate at surge is approximately 5% and that for pressure ratio at surge is approximately 1%, so that the pressure ratio at surge can be predicted more accurately. One reason for this is believed to be that the pressure ratio at surge has a strong correlation with impeller tip Mach number. This is clearly shown by the relationships, illustrated in **Fig. 7**, that exist between the impeller tip Mach number, and volumetric flow rate and pressure ratio at surge in the training data used in this study. Although different data were used, when the volumetric flow rate at surge was predicted with Tamaki's[25] one-dimensional physical model, the MRE was approximately 10%, so that prediction was relatively accurate, almost achieving the target MRE value of 5%.

Table 4 MRE of prediction of volumetric flow rate at surge and pressure ratio at surge for test data

Product type	Volumetric flow rate at surge (%)	Pressure ratio at surge (%)
Whole of verification data	5.02	1.38

2.3 Minimization of pressure loss of LPT (Low-Pressure Turbine) airfoil with DQN

To verify the performance of the DQN, we first performed a numerical experiment for shape optimization of a NACA (National Advisory Committee for Aeronautics) blade, which is a blade profile used for wind turbine generators and airplanes, using lift-to-drag ratio as the objective function. Following this, we performed shape optimization of a LPT (Low-Pressure Turbine) blade using pressure loss as the objective function. We used Chainer RL ver. 0.3.0[26] for DQN training.

2.3.1 Maximization of lift-to-drag ratio of NACA blade

An NACA blade has a blade profile, such as NACA6410, whose name contains numerical values that represent parameters which uniquely determine its shape. The NACA 4-digit series contains three shape parameters: maximum camber normalized by chord length, maximum camber position, and maximum blade thickness. The NACA 5-digit series contains the centerline profile in addition to these parameters. **Figure 8** shows an example of the NACA 4-digit series. In this study, the blade profile was fixed and the angle of attack, at which the fluid hits the leading edge of the blade, was optimized with a DQN so that the lift-to-drag ratio, which is larger-the-better, was maximized. The agent selected whether to increase or decrease the angle of attack, and this increase/decrease was defined as action a_t. The angle of attack was changed in increments of 1 degrees from 0 to 40 degrees. The agent was designed to receive a 400 × 400 px pressure contour diagram as state s_t and the scalar value of the lift-to-drag ratio as reward r_t. The contour diagram was output by creating a mesh having an appropriate external environment for the angle of attack, and performing two-dimensional steady incompressible CFD calculation and post-processing. In order for the agent to determine

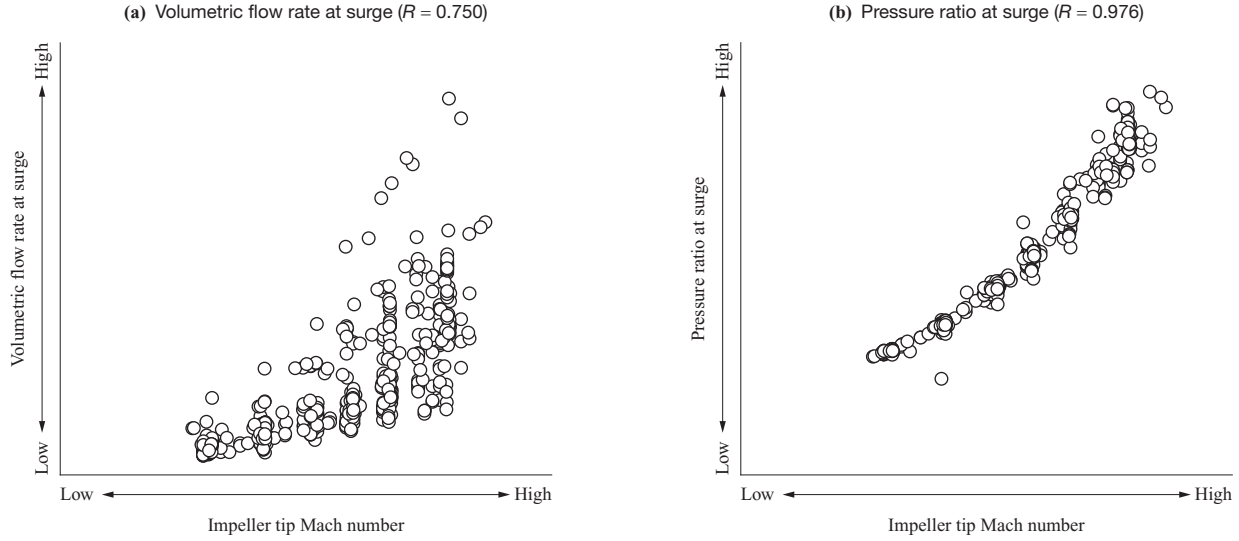
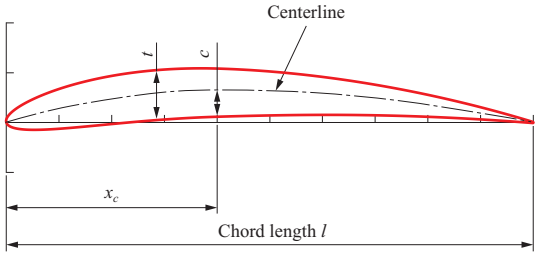


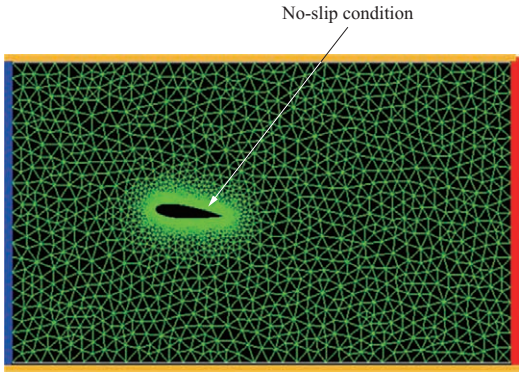
Fig. 7 Correlation coefficient R between impeller tip Mach number and prediction targets for training data



(Note) NACA *XYZZ*
$X = c/l \times 100$: Maximum camber
$Y = x_c/l \times 10$: Maximum camber position
$ZZ = t/l \times 100$: Maximum blade thickness

Fig. 8 Schematic image of NACA 4-digits

whether to increase or decrease the angle of attack based on the pressure contour diagram, the CNN specialized for image recognition described in **Section 2.1.2** was used to approximate the Q function. For CFD, OpenFOAM ver. 1706[27] was used, and the boundary conditions are shown in **Fig. 9**. The inlet boundary was set to a flow rate of 1 m/s, the outlet static pressure to 0 Pa, the blade surface to a non-slip condition, and the upper and lower boundaries of the analytical range to a slip condition, with a k-ε model being used for turbulence. In addition, the chord length and fluid density were set to 1 m and 1 kg/m^3, respectively, and training with the DQN was performed under three conditions, i.e., fluid viscosities of 1.0×10^{-3}, 1.0×10^{-5}, 1.0×10^{-7} Pa·s, so that *Re* (Reynolds number) was 10^3, 10^5, and 10^7. A flow field with an *Re* of 10^3 has laminar flow, for which it is not appropriate to use a k-ε model, and, in addition, in a separation region with an angle of attack of 20 degrees or more, a turbulence model must be selected which is appropriate for the size of vortex generated and grid resolution. However, since the purpose of this study is to verify the performance of the DQN, we determined the grid resolution and turbulence model in accordance with the angle of attack at which the lift-to-drag ratio is maximized, i.e., 5 to 20 degrees, in order to simplify the system. In the training, the initial angle of attack was determined randomly, and the agent changed the angle 50 times for 1 set, 400 sets



(Note) : Inlet boundary 1 m/s
: Outlet static pressure 0 Pa
: Slip condition

Fig. 9 Boundary conditions

being performed. As a reward, $r_t = +1$ was given when the lift-to-drag ratio increased after the angle of attack was changed, and $r_t = -1$ was given when it decreased. Here, as numerical experiments, we performed four experiments with different combinations of training and verification processes, as shown in **Table 5**, and evaluated the optimization performance of the DQN. With regard to the four numerical experiments shown in **Table 5**: in No. 1, the same 4-digit series was used for training and verification but different blade shapes were used; in No. 2, different blade series were used for training and verification; in No. 3, different blade shapes and *Re* values were used for training and verification; and in No. 4, training was performed under low-speed and high-speed conditions, and verification was performed under middle-speed conditions.

Figure 10-(a) shows the transition from the initial to the converged solution during verification for one of the blade shapes in No. 1. From the initial condition with an angle of attack of 37 degrees, the angle gradually decreased and converged to 6 degrees. **Figure 11-(a)** shows the pressure contours for the initial and converged solutions illustrated in **Fig. 10-(a)**. In the initial solution, the pressure is lower at the leading edge of the blade, and separation occurs. In the converged solution, however, the low-pressure area is smaller, and it is presumed that the agent detected the separation indicated by the pressure contours and took action to decrease the angle of attack. **Figure 12** shows the transition of action value function Q_θ for one of the sets during the training in No. 1, with the horizontal axis indicating the number of actions taken in the set. Before the number of actions reached around 30, the action value for decreasing the angle of attack was higher than that for increasing it. This shows that the agent clearly judged that it should take action to decrease the angle of attack because the initial angle was large. After the number of actions reached 30, the difference between the action values for increasing and decreasing the angle was small, meaning that the converged solution had almost been reached and the angle of attack stopped decreasing.

To evaluate the performance of the DQN in Nos. 1 to 4, the lift-to-drag ratio at the angle of attack at which the agent's actions converged was divided by the optimal lift-to-drag ratio within the search range of the agent. This calculation was performed for 20 blade shapes, and the average values for Nos. 1 to 4 are shown in **Fig. 13**. The larger the performance value shown on the vertical axis is, the higher the DQN's performance is. In No. 2, the NACA 5-digit series was used in verification, but *Re* was not changed, so it can be

Table 5 Conditions of numerical experiments

No.	Training		Verification	
	NACA	*Re*	NACA	*Re*
1	4-digit series	10^3	4-digit series	10^3
2	4-digit series	10^3	5-digit series	10^3
3	4-digit series	10^3	4-digit series	10^5
4	4-digit series	10^3, 10^7	4-digit series	10^5

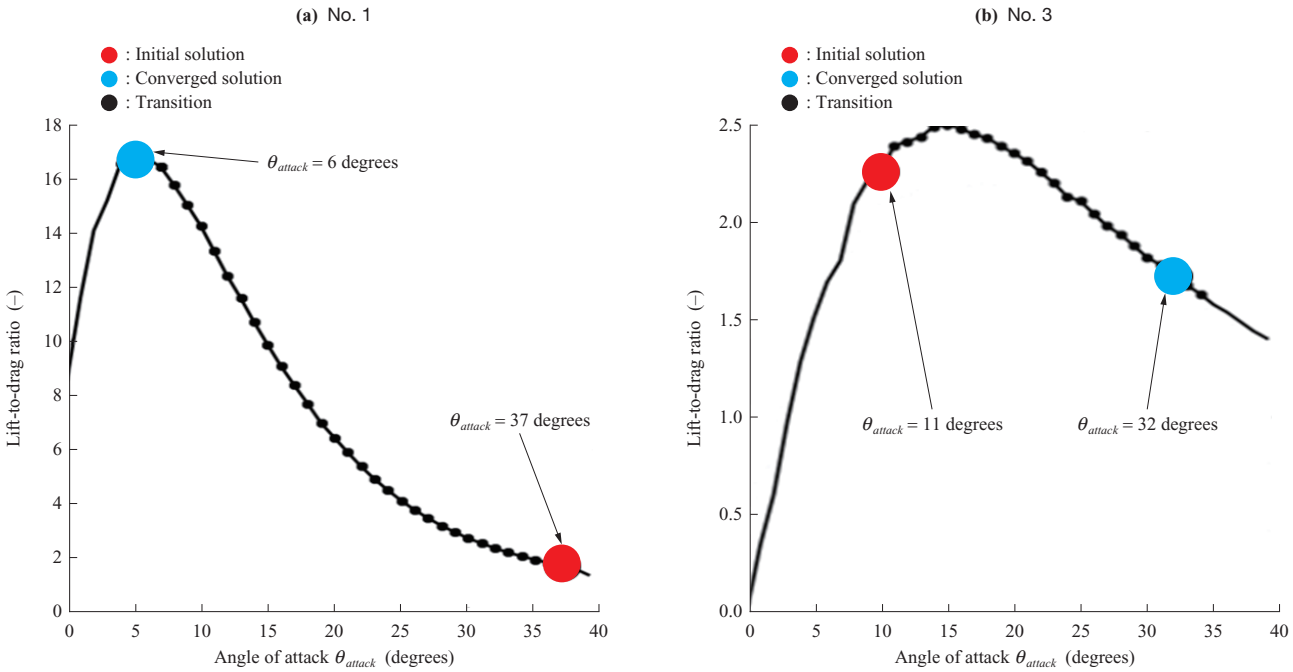


Fig. 10 Transition of lift-to-drag ratio from initial solution to converged solution

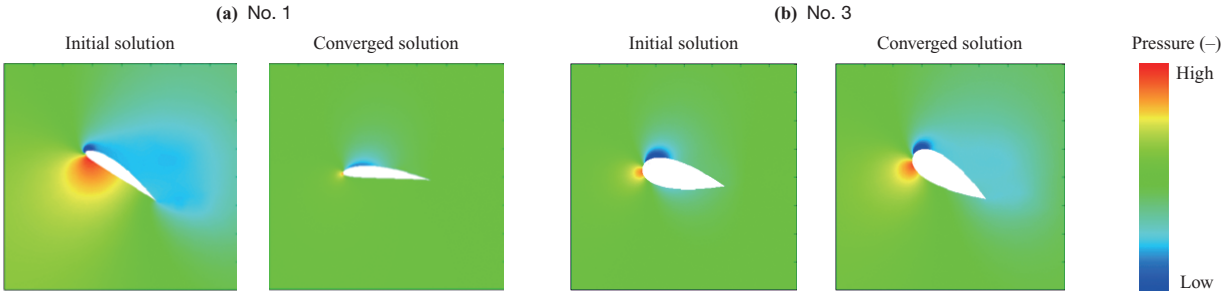


Fig. 11 Pressure contour map of initial and converged solutions in Fig. 10

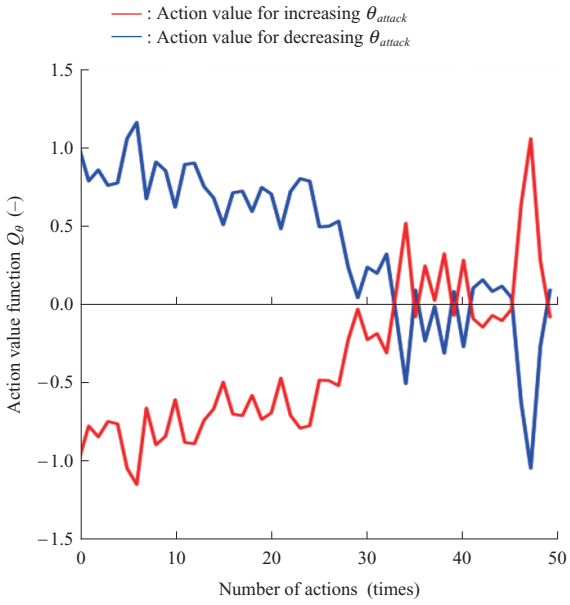


Fig. 12 Transition of Q function of No. 1

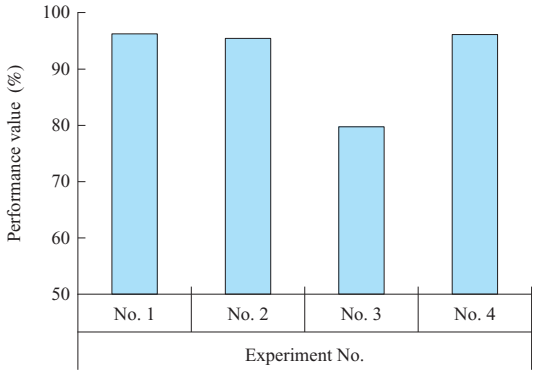


Fig. 13 Performance comparison among each numerical experiment

seen that the performance for No. 2 is almost the same as that for No. 1. In No. 3, where *Re* was changed in verification, the performance value is approximately 17 points smaller than for Nos. 1 and 2. This is presumably because, since *Re*

was different, some tendency that was not observed in training appeared in the pressure contours obtained by verification, showing that performance cannot be properly optimized if the flow field changes. **Figure 10-(b)** shows the transition from the initial to the converged solution during verification for one of the blade shapes in No. 3, and **Fig. 11-(b)** shows the pressure contours of the initial and converged solutions illustrated in **Fig. 10-(b)**. The optimal angle of attack for this blade shape is 15 degrees, but **Fig. 10** shows that the angle converged to 32 degrees. In addition, **Fig. 11-(b)** shows that the separation at the leading edge observed in the initial solution remains in the converged solution. At the same time, it can be seen from **Fig. 13** that the performance value for No. 4 — which even for an *Re* not used in training is within the interpolation range of the training conditions — is almost the same as those for Nos. 1 and 2. From the above, it is presumed that a trained agent can be reused by training it with a DQN again under a wider range of conditions.

2.3.2 Minimization of pressure loss of LPT airfoil

Using PACK B[28] — which is a blade shape used in low-pressure turbines for aircraft engines — as a basis, we performed shape optimization to minimize pressure loss. We used four design variables, these being the camber length and blade thickness at the 1/3 and 2/3 positions of the chord length, as shown in **Fig. 14**. Here, the flow field conditions were fixed and the four design variables were optimized with a DQN so that the pressure loss, which is smaller-the-better, was minimized under the constraint conditions. The agent was designed to receive a 400 × 400 px Mach number contour diagram as state s_t. The contour diagram was output by creating a mesh having an appropriate external environment for the design variables, and performing two-dimensional steady incompressible CFD calculation and post-processing. In order for the agent to approximate the Q function, a CNN was used in which action a_t was taken that changes the four design variables based on the Mach contour diagram. If a constraint condition is applied to the DQN, it is necessary to consider both the objective function and constraint condition when defining the reward. Here, the objective function was pressure loss, and the constraint range was set to an outlet angle of −58.3 ± 0.2 degrees. In addition, as a reward, the evaluation function f represented by Equation (11) was defined, and $r_t = +1$ was given when the evaluation function increased, and $r_t = -1$ when it decreased.

$$f = -cP_{\text{drop}} - \left|\theta_{\text{const}} - \theta\right| \quad \cdots\cdots(11)$$

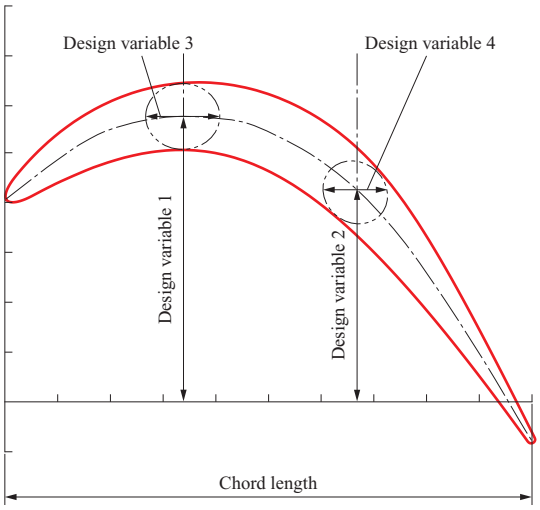
In this equation, P_{drop} is pressure loss [−], θ_{const} (= −58.3°) is the constraint condition outlet angle, and θ is the outlet angle obtained by CFD. The first term in Equation (11) represents the objective function and the second term the constraint condition, and these are weighted by a constant c (= 500). Pressure loss P_{drop} is calculated by the following equation:

$$P_{\text{drop}} = \frac{P^T_{\text{Inlet}} - P^T_{\text{Outlet}}}{P^T_{\text{Outlet}} - P^S_{\text{Outlet}}}. \quad \cdots\cdots(12)$$

In this equation, P^T_{Inlet} is the total pressure at the inlet, P^T_{Outlet} is the total pressure at the outlet, and P^S_{Outlet} is the static pressure at the outlet, each of which can be obtained by CFD. In the training, the initial shape was determined randomly, and 200 sets of training were performed, with the shape being changed 50 times in each set.

Figure 15 shows the transition to the converged solution when an initial solution was optimized using a trained agent (**-(a)**) and the distribution of converged solutions for 15 initial solutions (**-(b)**). The horizontal axis shows outlet angle, which is the constraint condition, and the vertical axis shows pressure loss, which is the objective function. The optimal solution is the one for which pressure loss is minimum within the constraint range shown in the figure. In **Fig. 15**, the points for approximately 12 000 shape candidates are also plotted. From **Fig. 15-(a)**, it can be seen that the solution converges within the constraint range. In addition, **Fig. 15-(b)** shows that almost all the randomly created initial shapes converge to optimal solutions within the constraint range.

Figure 16-(a) shows the Mach number contours for the initial and converged solutions in **Fig. 15-(a)**. In general, when designers design a blade based on the Mach number contour, they focus on areas ① and ② in **Fig. 16-(a)**, and evaluate pressure loss based on the separation at the back-side trailing edge, and outlet angle based on the velocity distribution at the leading edge. Therefore, so as to understand which part of the Mach contour received as a state the agent focuses on in order to determine its actions, we examined the feature map (image filtered in the convolutional layer) for the contour diagram input into the CNN which approximates the Q function. For the Mach number contour in **Fig. 16-(a)**, multiple feature maps can be obtained from the CNN. Two maps, A and B, were selected from among these, and are shown in **Figs. 16-(b)** and **-(c)**. To improve visualization of the feature maps, the blade shape and calculation range are shown for convenience. The bright areas in each feature map



(Notes) Design variable 1 : Camber length at 1/3 position of chord length
Design variable 2 : Camber length at 2/3 position of chord length
Design variable 3 : Blade thickness at 1/3 position of chord length
Design variable 4 : Blade thickness at 2/3 position of chord length

Fig. 14 Design variables of LPT airfoil

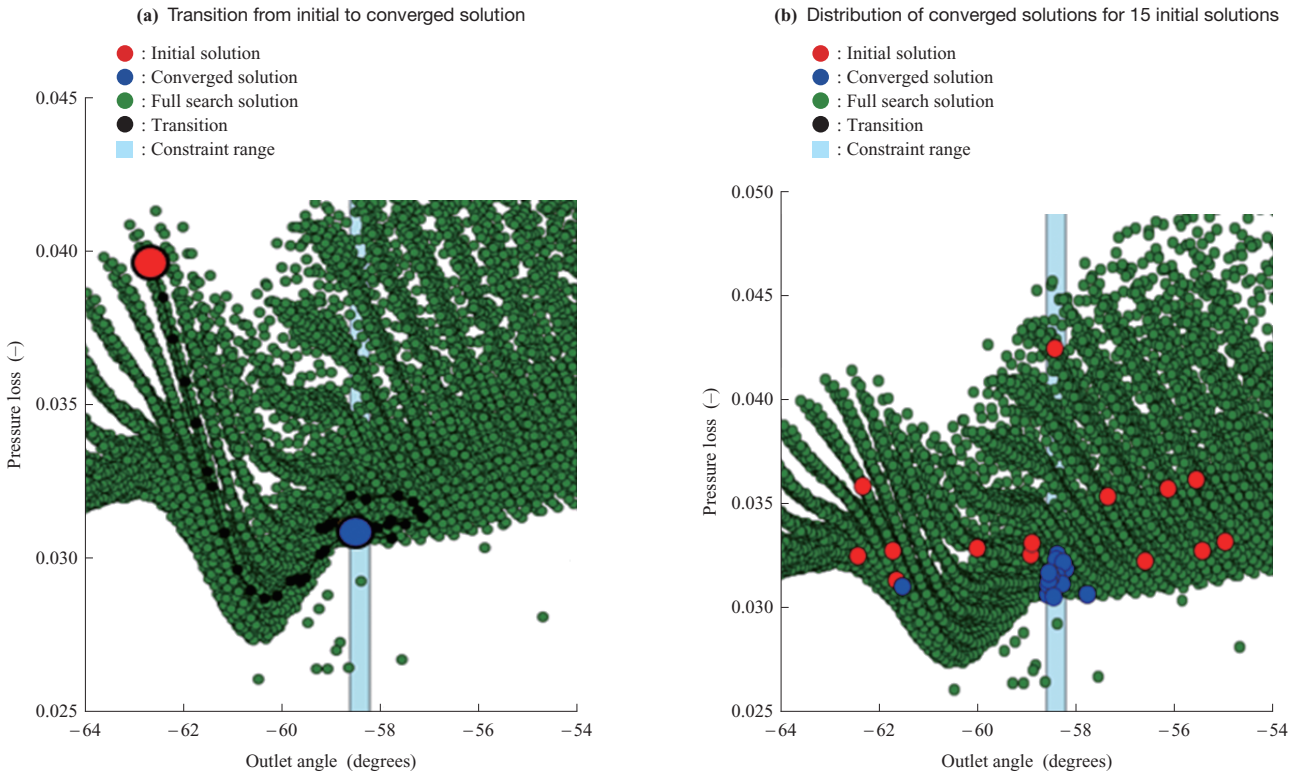


Fig. 15 Initial solutions and its converged solutions

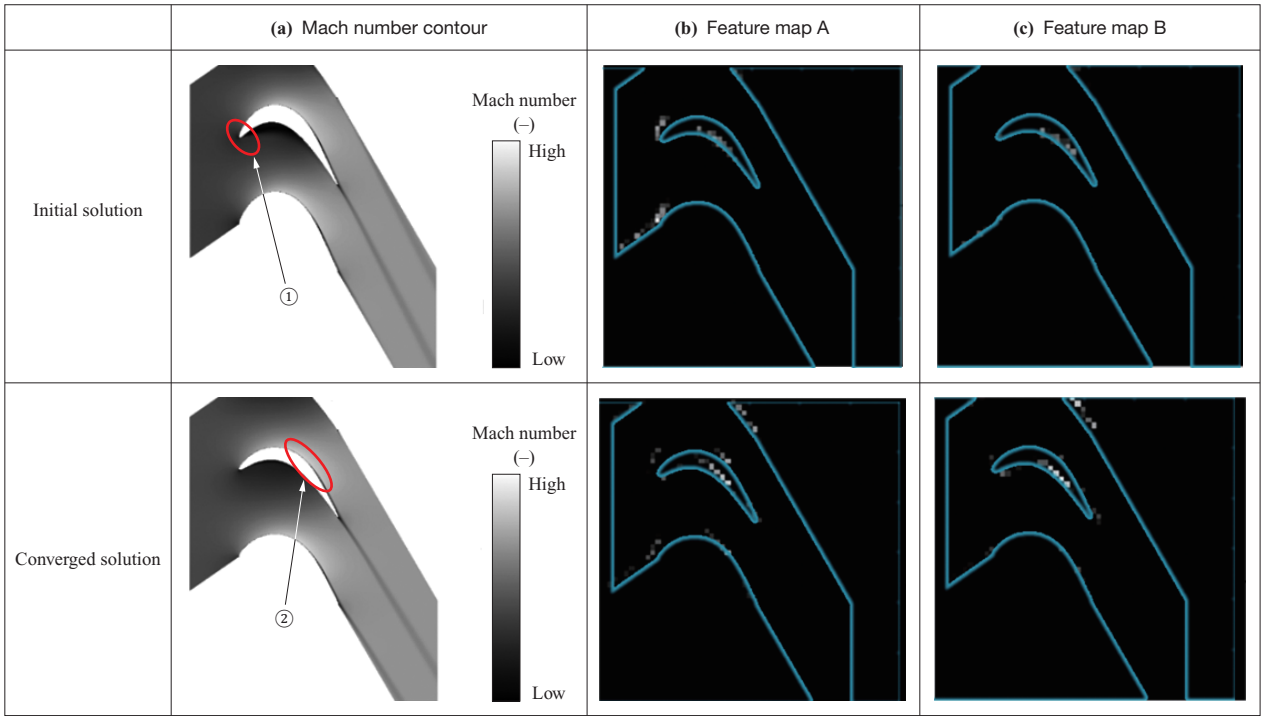


Fig. 16 Mach number contour and feature maps of initial and converged solutions of Fig. 15-(a)

indicate characteristic areas in the Mach number contour extracted by filtering. From the feature maps in **Figs. 16-(b)** and **-(c)**, it can be seen that features appear at the leading and back-side trailing edges in the initial solution but disappear in the converged solution. From this, it is presumed that the agent took action to eliminate the features appearing in each area. The areas where features appear and then disappear correspond to the areas ① and ② that designers focus on, as shown in **Fig. 16-(a)**, so it can be said that the agent and designers focus on the same points.

3. Conclusion

In order to develop a performance prediction technique using machine learning and design solution search technique using reinforcement learning that complement the unknown parts of the relevant physical phenomena, we investigated vehicle turbocharger compressors and LPT airfoils — which constitute turbomachinery, predicted volumetric flow rate with a DNN using the calculated values obtained by one-dimensional CAE, and minimized pressure loss through a combination of CAE and a DQN.

In predicting the volumetric flow rate at surge of a compressor using a DNN, we successfully developed a prediction model at low cost by utilizing past measurement data for training, and, with unknown data, predicted pressure loss with a relative error of approximately 5%. Because the input parameter values used for prediction are not included in the training data and prediction accuracy decreases under extrapolated conditions, it is important to expand the training data and reduce the extrapolation region as much as possible. The prediction model developed through this technique enables high-speed low-cost prediction and is expected to be used in the early stages of the design process.

With regard to optimization using a DQN, we have confirmed that the agent can be properly trained within the conditions learned through numerical experiments. It has also been discovered that even outside the range of the learned conditions, if those conditions are within the interpolation range, then optimization with a DQN is possible. This suggests that as long as training is performed such as to cover a sufficient design range, appropriate optimization is possible for various problems. In minimizing the pressure loss of an LPT airfoil, we confirmed that optimization can be achieved by setting an appropriate reward for a problem that has a constraint condition. Since this technique allows any reward to be defined, it can also be applied to other problems. Through feature maps of the convolutional layer of a CNN that approximates the Q function, we confirmed that the agent and designers focus on the same points. As was done in this study, feeding back the design points learned by machine learning to the designers may enable them to verify the validity of the machine learning model and obtain new findings.

REFERENCES

(1) K. Deb : Multi-objective Optimization using Evolutionary Algorithm, John Wiley & Sons, 2005

(2) R. H. Myers and D. C. Montgomery : Response Surface Methodology: process and product optimization using designed experiments, second edition, A Wiley-Interscience publication, 2002

(3) S. Urolagin, K. V. Prema and N. V. Subba Reddy : Generalization Capability of Artificial Neural Network Incorporated with Pruning Method, International Conference on Advanced Computing, Networking and Security, 2011, pp. 171-178

(4) G. E. Hinton, S. Osindero and Y. W. Teh : A Fast Learning Algorithm for Deep Belief Nets, Neural Computation, Vol. 18, 2006, pp. 1 527-1 554

(5) V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller : Playing Atari with Deep Reinforcement Learning, NIPS Deep Learning Workshop 2013

(6) U. Kesgin : Genetic Algorithm and Artificial Neural Network for Engine Optimisation of Efficiency and NO_x emission, Fuel, Vol. 83, Iss. 7-8, 2003, pp. 885-895

(7) O. Ozener, L. Yuksek and M. Ozkan : Artificial Neural Network Approach to Predicting Engine-out Emissions and Performance Parameters of a Turbo Charged Diesel Engine, Thermal Science, Vol. 17, No. 1, 2013, pp. 153-166

(8) B. Liu, C. Zhao, F. Zhang, T. Cui and J. Su : Misfire Detection of a Turbocharged Diesel Engine by using Artificial Neural Networks, Applied Thermal Engineering, Vol. 55, No. 1, 2013, pp. 26-32

(9) E. Mese and D. A. Torrey : An Approach for Sensorless Position Estimation for Switched Reluctance Motors using Artificial Neural Networks, IEEE Transactions on Power Electronics, Vol. 7, Iss. 1, 2002, pp. 66-75

(10) U. Atici : Prediction of the Strength of Mineral Admixture Concrete using Multivariable Regression Analysis and an Artificial Neural Network, Expert Systems with Applications, Vol. 38, Iss. 8, 2011, pp. 9 609-9 618

(11) X. Glorot, A. Bordes and Y. Bengio : Deep Sparse Rectifier Neural Networks, In Proc. 14th International Conference on Artificial Intelligence and Statistics 2011, pp. 315-323

(12) A. Maas, A. Hannun and A. Ng : Rectifier nonlinearities improve neural network acoustic models, ICML 2013

(13) D. P. Kingma and J. Ba : Adam: A Method for Stochastic Optimization, Proceedings of the 3rd ICLR 2015

(14) N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov : Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research, Vol. 15, 2014, pp. 1 929-1 958

(15) D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel and D. Hassabis : Mastering the Game of Go with Deep Neural Networks and Tree Search, Nature, Vol. 529, 2016, pp. 484-489

(16) M. Vecerik, T. Hester, J. Schölz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothorl, T. Lampe and M. Riedmiller : Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards, arXiv:1707.08817v2, (2018)

(17) A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.

M. Allen, V. D. Lam, A. Bewley and A. Shah : Learning to Drive in a Day, arXiv:1807.00412v2, 2018
(18) R. S. Sutton and A. G. Barto : Reinforcement Learning: An Introduction, The MIT Press, 2017
(19) A. Krizhevsky, I. Sutskever and G. E. Hinton : ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012, pp. 1 097-1 105
(20) C. Bishop : Pattern Recognition and Machine Learning, Springer, 2006
(21) J. Snoek, H. Larochelle and R. Adams : Practical Bayesian Optimization of Machine Learning Algorithms, NIPS 2012, pp. 2 951-2 959
(22) Chainer : A flexible framework for neural networks, < https://chainer.org >, accessed 2018-10-30
(23) GPyOpt : < https://sheffieldml.github.io/GPyOpt/index.html >, accessed 2018-10-30
(24) Scikit-learn : machine learning in python, < http://scikit-learn.org/stable/ >, accessed 2018-10-30
(25) H. Tamaki : Effect of Piping Systems on Surge in Centrifugal Compressors, Journal of Mechanical Science and Technology, Vol. 22, Iss. 10, 2008, pp. 1 857-1 863
(26) Chainer RL : < https://github.com/chainer/chainerrl >, accessed 2018-10-30
(27) OpenFOAM : < http://www.openfoam.org >, accessed 2018-10-30
(28) J. P. Lake : Flow Separation Prevention on a Turbine Blade in Cascade at Low Reynolds Number, PhD Thesis, Air Force Institute of Technology, 1999

INFORMATION

Thank you very much for reading the article of IHI ENGINEERING REVIEW.
We hope you will take a look at other articles.

Webpages of our journals on technology are as follows:

Journal of IHI technologies (in Japanese)

IHI ENGINEERING REVIEW (in English)

Vol. 54 No. 2

Contents page of Vol.54 No.2

Our corporate website introduces our technology categorized according to social issues: “IHI Challenges with Society”. The articles of IHI ENGINEERING REVIEW are also provided there. We would appreciate it if you would visit our website.

IHI Challenges with Society

Technologies supporting IHI Products