

# Cooling channel designs of a prismatic battery pack for electric vehicle using the deep Q-network algorithm

Y.T. Kim<sup>a</sup>, S.Y. Han<sup>b,\*</sup>

<sup>a</sup> Department of Convergence Mechanical Engineering Master's degree, University of Hanyang, 222, Wangsimni-ro, Seongdong-gu, Seoul, Korea

<sup>b</sup> School of Mechanical Engineering Professor, University of Hanyang, 222, Wangsimni-ro, Seongdong-gu, Seoul, Korea

## ARTICLE INFO

### Keywords:

Cooling channel design  
Deep Q network (DQN)  
Electric vehicle  
Battery cooling  
CFD analysis

## ABSTRACT

In this study, using the deep Q-network (DQN) algorithm, which is suitable for cooling channel design, a design method that satisfies the specified target inputs, namely, maximum temperature, average temperature, temperature standard deviation, and pressure drop, was proposed. The cooling channel aims to design this shape. The agent designs this shape through grid environment experience and obtains a reward through the analysis results of the generated shape. Finally, one obtains the maximum reward through the learned policy. With the proposed design method, it was possible to obtain the optimal cooling channel and the maximum reward for three examples. The final DQN results were verified for validity by comparing them with the Ansys results. Computational fluid dynamics (CFD) analysis requires high-quality mesh generation and selection of an analysis technique suitable for the problem and high proficiency. Therefore, it is expected that the proposed method will not only shorten the calculation time but also design the cooling channel according to various conditions.

## 1. Introduction

Electric vehicles use a large amount of power as electricity and must run for as long as possible on a single charge, so they require a high-capacity battery [1]. High-capacity batteries are packaged by connecting many battery cells in a modular format in a small space inside the vehicle. Therefore, a battery pack cooling system to manage the heat generated by the battery is very important [2].

Lithium-ion batteries, which are used most often for electric vehicles, are the most suitable batteries for electric vehicles [3–5]. In addition, the recommended operating temperature range for lithium-ion batteries is 20 °C to 40 °C [6–10]. If the battery pack cooling system is not functioning properly, the abnormal temperature rise of the battery pack can shorten the battery life and cause thermal runaway [11]. In addition, since increasing the battery temperature can decrease battery efficiency, it is necessary to maintain the battery temperature uniformly through the battery pack cooling system to prevent the battery pack from overheating [12].

The battery system of electric vehicles is mainly used to efficiently utilize a narrow space through a pack type in which prismatic battery cells are stacked for high capacity and light weight [13]. An aluminium

cooling plate with a built-in cooling channel is sandwiched between the stacked prismatic battery packs, and the heat transferred from the battery pack is cooled by the refrigerant flowing into the cooling channel. The aluminium cooling plate has a different temperature distribution depending on the pressure of the refrigerant flowing through the cooling channel, the temperature of the refrigerant, the position of the inlet and outlet, and the path. Therefore, to keep the temperature of the battery pack cooling system uniform, it is necessary to design an aluminium cooling plate with efficient cooling channels.

The battery cooling methods of a general electric vehicle can be divided into an air-cooling type and a liquid cooling type. The air-cooling method is widely used because of its low cost and natural and forced convection. Qian et al. [14] optimize the battery spacing to improve the heat dissipation and air-cooling performance of a cylindrical battery pack and to train an artificial neural network with the maximum temperature and temperature difference values obtained from CFD simulations. However, when the battery has a high discharge rate and is used in various environmental conditions, the liquid-cooled method is more efficient in terms of cooling performance than the air-cooled method because of its higher thermal conductivity [15].

As a method for optimizing an aluminium cooling plate that can

Abbreviations: DQN, Deep Q network.

\* Corresponding author.

E-mail address: [syhan@hanyang.ac.kr](mailto:syhan@hanyang.ac.kr) (S.Y. Han).

maintain the temperature difference between battery cells at an appropriate temperature, the best multi-pass serpentine flow-field (MPSFF) model was presented by Yu et al. [16] by comparing the cooling performance according to six MPSFFs at the inlet and outlet fixed to the cooling plate. Jarrett and Kim [17] proposed a process for optimizing the thickness and position of the cooling channel in the channel design selected using the Latin hypercube sampling method based on the above paper. In addition, within the same objective function, the effect on the optimal cooling plate design was studied according to the conditions of coolant flow rate, battery calorific value, and nonuniform temperature distribution [18]. Panchal et al. [19] proceeded to optimize the thickness and position of a given cooling channel in a mini-channel cooling plate placed in a prismatic lithium-ion battery cell using experimental and numerical techniques. In addition, the battery discharge rate and inlet temperature were changed and verified through analysis and experiments according to the operating conditions, but only the cooling performance for the given pass patterns was compared as in the study above. However, since the above existing methods only compared the cooling performance for the given serpentine pass patterns, various other routes cannot be considered [16,17]. In addition, although the  $x$  and  $y$  dimensions of a given cooling channel were parameterized and optimized regarding the channel of the cooling plate, an optimal cooling channel that is difficult to be applied to an actual system was derived [17,18].

CFD analysis is an important research field in the field of heat and fluids, and the verification method through analysis and experiment requires a lot of trial and error as well as time and money [20]. Additionally, existing optimization techniques require a new definition of the problem when the design domain or constraints change, and optimization calculations must be redone each time. Further, CFD analysis requires a high level of proficiency because the results vary depending on the quality and number of meshes. As a way to solve the above problem, it is necessary to consider various cooling channels that are suitable for these use conditions by applying deep learning. Recently, research to solve the heat transfer problem using machine learning (ML) has been proposed. Hobold et al. [21] used 10,000 boiling water images to train the ML model. After learning how to relate boiling water images to heat fluxes, they made near-accurate predictions. And Kwon et al. [22] used a random forest algorithm to predict a higher-order nonlinear heat transfer problem, that is, the convective heat transfer coefficient of a cooling channel with variable rib roughness. Both gave near-accurate predictions, but the drawback is that they require training data.

Reinforcement learning (RL) is a branch of ML in which an agent defined in a given environment chooses actions and sequences of actions that maximize rewards through actions that can be selected in the current state [23]. Because RL learns policies and derives results based on received rewards, it actively collects data based on the agent's behaviour rather than the designer's experience; thus, RL is widely used for control [24], recommendation [25], and optimization [26,27] among ML techniques. Q-learning, one of the most representative RL methods, finds a Q-value to determine an optimal policy from a Q-function that expresses quality [23,28]. Q-learning can solve simple RL problems, but as the size of the environment increases, learning becomes more complex and difficult. To solve the above problems, deep Q-learning was developed by applying deep learning to Q-learning.

Deep Q-learning, which was first developed in Google's DeepMind, is also called a deep Q-network (DQN) because the Q-function is composed of a deep neural network (DNN). In DQN, DNN is used to approximate the optimal Q-value function, and results can be derived by learning various design parameters. Therefore, the use of a DQN makes it possible to solve complex and difficult RL problems such as Atari games and path finding [29–31].

In this research, RL using a DQN was performed to find various channels for the cooling plate that satisfy the target constraint in the prismatic battery pack cooling system used for electric vehicles. As for the DQN used, the agent observes the grid environment from the inlet

point, and when the agent arrives at the outlet, it performs an analysis through the CFD analysis environment and trains according to the reward constraint for the obtained result. Additionally, the agent designs the channel of the cooling plate that satisfies the maximum temperature, average temperature, temperature standard deviation, and pressure drop within the range of target conditions. A method for deriving a new cooling plate channel according to the learned policy was proposed and verified even if the locations of the inlet, outlet, and domain size were changed through learning.

## 2. Methods

### 2.1. Q-learning

Unlike supervised learning and unsupervised learning, RL is a ML technique that allows an agent to learn by itself, and the agent learns an action to maximize their reward through trial-and-error in a dynamic environment [32]. The agent collects data by taking action in a given environment and gets a reward for the action taken in the current state.

Fig. 1 shows a component of RL, and the decision-making method through the interaction between the agent and the environment is called the Markov decision process (MDP) [33–34]. MDP can be expressed as a tuple formula of  $(S_t, A_t, P_{SA}, R, \gamma)$ , where  $S_t$  is the state,  $A_t$  is the action,  $P_{SA}$  is the state transition probability,  $R$  is the reward, and  $\gamma$  is the discount factor.

First, among the components of MDP, reward is the only information that the agent can learn, and it can be obtained from the environment and can be expressed by Eq. (1). That is, the reward is  $R_t(s, a)$  obtained when the state is  $S_t = s$  and the action is  $A_t = a$  at time  $t$  means the expected value  $E$ . Here,  $E$  is a kind of average value, considered as the predicted value rather than an exact value, and  $R_{t+1}$  is the reward for the action that can be obtained in the next state.

$$R_t(s, a) = E[R_{t+1} | S_t = s, A_t = a] \quad (1)$$

Among the components of MDP, the state transition probability  $P_{SA}$  is the probability of reaching another state  $S_{t+1}$  when the agent takes an action ( $A_t$ ) in a certain state ( $S_t$ ) of the environment and is expressed as Eq. (2).

$$P_{SA} = P[S_{t+1} = s | S_t = s, A_t = a] \quad (2)$$

In RL, the agent needs a policy ( $\pi$ ) to perform an action, and the policy contains information about the actions that the agent can perform in all states of the grid environment and is expressed as Eq. (3).

$$\pi(a|s) = P[A_t = a, S_t = s] \quad (3)$$

That is,  $\pi(a|s)$  is the probability of acting  $A_t = a$  among the possible actions when there is an agent in  $S_t = s$  at time  $t$ , and the goal is to experience the state and action based on the probability and establish an optimal policy ( $\pi^*$ ).

To find the optimal policy, values for a specific state and specific action are determined, and the values are determined by the state value function  $v_\pi(s)$  and the action value function  $Q_\pi(s, a)$ .  $v_\pi(s)$  is an expected value that judges the value of the state and is expressed as Eq. (4).

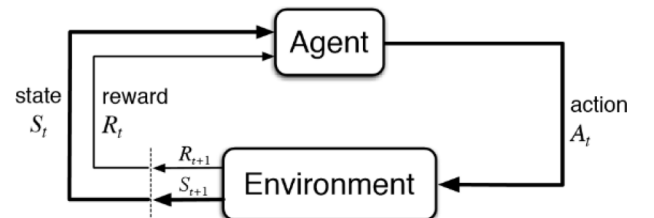


Fig. 1. Reinforcement learning model.

$$v_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma \bullet v_{\pi}(S_{t+1}) | S_t = s] \quad (4)$$

Eq. (4) is the Bellman expectation equation, and the value of  $s$  expected through  $\pi$  at  $t$  is expressed as the sum of the rewards to be received in the future [28].  $\gamma$  in Eq. (4) is a discount factor, and since the present reward has a different value from the future reward, it can be seen that the future value is converted into the present value. The action value function is  $Q_{\pi}(s, a)$ ; that is, the expected value of the Q-function, is expressed in the form of an action added to the state value function as in Eq. (5) [28].

$$Q_{\pi}(s, a) = E_{\pi}[R_{t+1} + \gamma \bullet Q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] \quad (5)$$

If  $v_{\pi}(s)$  and  $Q_{\pi}(s, a)$  are continuously updated, a converged value can be obtained, but since it is an expected value for the currently acting policy, it is not a value function for the optimal policy. Therefore, it is necessary to find the optimal value function in the optimal policy, and it can find the policy giving the highest value among all the policies through the max function, which is expressed by Eqs. (6) and (7).

$$v^*(s) = \max_{\pi}[v_{\pi}(s)] \quad (6)$$

$$Q^*(s, a) = \max_{\pi}[Q_{\pi}(s, a)] \quad (7)$$

The Bellman optimal equation for the state and behaviour through the max function is expressed by Eqs. (8) and (9), and in Eq. (9),  $a'$  represents the action that maximizes the Q-value in  $s'$ .

$$v^*(s) = \max_a E[R_{t+1} + \gamma \bullet v^*(S_{t+1}) | S_t = s, A_t = a] \quad (8)$$

$$Q^*(s, a) = E \left[ R_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \middle| S_t = s, A_t = a \right] \quad (9)$$

Eq. (10) updates the Q-function at every time step, and  $\alpha$  is the learning rate, which refers to the shift step of the differential gradient, and is a value used when updating the Q-function. If the learning rate value is small, the learning time is slowed down because the gradient movement interval per step is small; conversely, if it is large, the interval of movement of the gradient per step becomes large causing the data to deviate chaotically and not converge to the lowest point, thus diverging.

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \quad (10)$$

In Q-learning, the agent explores the given environment and learns enough and then creates a Q-table that is used as the agent policy. The agent solves the Q-learning problem using the Q-table where all the states of the environment are stored, but there is a limit to saving the Q-table when the number of states is too many. In this study, this problem was solved through the DQN applied with a DNN.

## 2.2. Deep Q-network

DQN can solve the problem of Q-learning because Q-values are stored as a model called the weight ( $\theta$ ) rather than being learned in a table format. However, simply attaching a DNN cannot effectively use a DQN. In RL, there is no label such as supervised learning or unsupervised learning, and because it learns as a reward for the agent actions over time, there is a high correlation between adjacent data, and incomplete learning occurs. These problems can be solved using the method of experience replay, which learns by randomly configuring the data stored in  $D_t = \{e_1, e_2, \dots, e_t\}$  according to the time step of the agent's experience  $e_t = (s, a, s', r)$  in a mini-batch as shown in Eq. (11) [35–36].

$$(s, a, s', r) \sim U(D) \quad (11)$$

Afterward, the correlation of input data through the mini-batch is reduced, making the main network  $Q(s, a; \theta)$  and target network  $\hat{Q}(s, a; \theta^-)$  independent networks. In the main network, the problem of

divergence of the target  $y_i$  is solved by predicting the correct answer through continuous learning and updating the target network at uniform intervals. Equation (12) is a loss function ( $L(\theta)$ ), which consists of a predicted value  $y_i$  and an actual value  $Q(s, a; \theta)$  as an error function, and the goal is to minimize it.

$$L(\theta) = E_{(s, a, s', r) \sim U(D)} [(y_i - Q(s, a; \theta))^2] \quad (12)$$

$$\text{where } y_i = r + \gamma \max_{a'} \hat{Q}(s, a; \theta^-) \quad (13)$$

To maximize the performance of DQN through this method, it is necessary to select an appropriate hyperparameter.

## 3. Design procedure of battery cooling channel

The maximum temperature ( $T_{max}$ ), average temperature ( $T_{ave}$ ), temperature standard deviation ( $T_{\sigma}$ ), and pressure drop ( $P_{fluid}$ ) of the cooling plate of an electric vehicle change according to the channel design, and the channel design for maintaining the temperature of the battery cell is varied. The optimal operating temperature of a battery cell in an electric vehicle is 288.15–308.15 K, and when the maximum  $T_{\sigma}$  between modules of the battery pack exceeds 5 K, the efficiency of the battery cell rapidly decreases [6–10]. Therefore, it is necessary to design a cooling channel that lowers  $T_{max}$ ,  $T_{ave}$ , and  $T_{\sigma}$  to maintain an appropriate operating temperature.

In this study, the deep learning model was verified through the prediction of the cooling channel by learning an environment of a certain size to design a cooling channel suitable for the conditions of use of the battery cell by applying DQN. In addition, comparative verification of the deep learning prediction results was performed through the commercial software Ansys CFD analysis.

Fig. 2 shows the RL process, and the environment for creating the DQN cooling channel includes a grid environment in which the agent explores the channel and a CFD analysis environment in which the temperature distribution of the cooling plate is calculated. In the training process, only when the agent arrives at the outlet can CFD analysis be performed to update the reward. Even in the case of the simple examples used in this study, the analysis takes a long time when using commercial analysis programs such as Ansys and COMSOL, so the time taken from mesh generation to CFD analysis results was shortened by using the CFD analysis environment openFOAM [37].

### 3.1. Grid environment

In this study, a cooling channel was designed by adding a CFD analysis environment during path finding, which is a representative example of RL. The training of RL is created through two environments, where the grid environment is the space that contains the agent action as

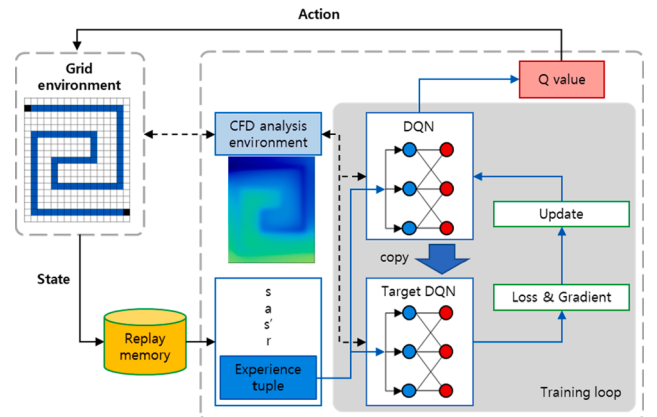


Fig. 2. Process of cooling plate design using DQN.

well as the shape information of the cooling plate and cooling channel. The actual cooling channel and grid environment of the cooling plate are three-dimensional (3D) with a constant thickness, and the total size of the grid environment becomes the total size of the cooling plate. The visit coordinates are exported as grid coordinate parameters visited by the agent and become the center coordinates of the cooling channel. In addition, the width of the cooling channel is set as a variable and freely changed within a certain range.

The size of the environment in RL greatly affects the learning results, and an environment without constraints greatly increases the number of cases for the agent action and experiences the environment through many episodes. This increases the probability that the agent cannot arrive at the outlet starting from the inlet, and the probability that the learning result does not converge is increased. Therefore, in this study, the size of the environment was downscaled as in Fig. 3 to 1/10 of the 3D CAD model, and the environment was constructed and learned. The reduced environment corresponds to a 3D CAD model of 10 mm  $\times$  10 mm per grid, and each time an agent performs an action, a channel is created in the 3D CAD model by 10 mm. Multiply the coordinate values (4 vertices) found in the grid environment by 10 to get the coordinates of the 3D CAD model. For example, starting from the starting point (0,0) and moving 1 space, the coordinates of the 4 points become (0,0), (1,0), (1,1), (0,1). Multiplying a coordinate value by 10 in a 3D CAD modelling program produces a rectangle having 4 coordinates (0,0), (10,0), (10,10), (0,10). Based on the coordinates visited in the grid environment, 3D modelling was carried out, and the thickness was set as a fixed variable with a fixed value of 0.5 mm. In this way, we can adjust the real scale of a desired cooling channel. In addition, if the conditions for the action are not given, the direction change is free, so the phenomenon of performing stepped actions similar to that shown in Fig. 4 occurs frequently.

In this study, two constraints were added to prevent such a stepped action phenomenon.

Fig. 4 in Section 3.1 shows how an agent moves from the top left to the bottom right. Action as shown in the figure results in a high pressure drop in the cooling channel. The pressure drop increases much more as the channel bends or lengthens more. Therefore, the action was restricted through a negative reward whenever the grid was visited. And agents are not permitted to visit external grids except for entrance and exit grids. Also, the previous state moved during the action cannot be revisited. In addition, the thickness range of the cooling channel shape used in this study is 10 to 16 mm, but in the grid environment created at 10 mm intervals, it is impossible to express on the screen in units of 11 to 1 mm; however, it is possible to express it in coordinates. To express this, the input of the actual value of the cooling channel thickness is input to the config.ini file. Then, in the grid environment, the value 10 mm corresponding to the minimum thickness range has 5 mm added symmetrically to the center coordinates of the cooling channel that arrives at the outlet through the input parameter script. For 10 mm or more, for every 1 mm increase, modeling proceeds with a value of 0.5 mm added symmetrically to 5 mm. In addition, the shape of the actual cooling

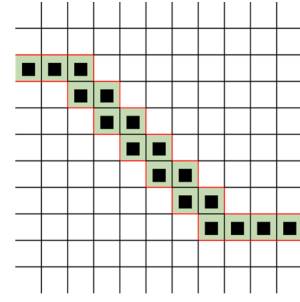


Fig. 4. Example of the stepped action.

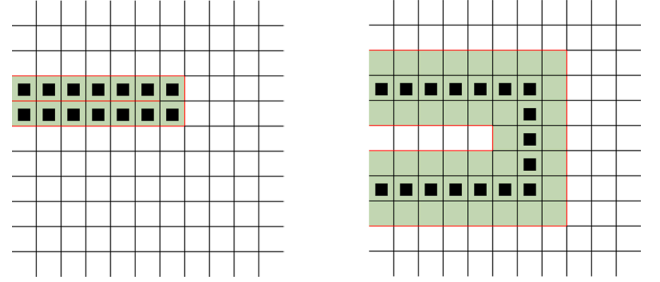


Fig. 5. (left) A 20 mm channel created by overlapping shapes when no spacing is given and (right) a cooling channel created with a grid spacing of 1 space.

channel was 10 mm or more, and as shown in Fig. 5 (right), the action was performed at least one space away from the already visited grid wall.

It is the 10 mm case where the agent starts at 4 and goes back to 5. At this time, the problem is that when visualized with 3D CAD, the 4th grid is not the inlet, but the 4th and 5th two grids are overlapped and it is created as a 20 mm inlet. Therefore, it was restricted not to visit one space on each left and right as shown in the right figure in Fig. 5. When designing a channel using DQN with these conditions, a cooling channel width was selected in consideration of the appropriate temperature of the cooling plate, and rewards were applied as shown in Table 1 to design a shape with a low pressure drop within the target temperature

Table 1  
Reward.

Per grid visit count	-1
Goal	+100
$T_{max}$ [K]	$303.151 \sim 318.15 = 0 \sim 100$
$T_{ave}$ [K]	$298.149 \sim 308.15 = 0 \sim 100$
$T_e$ [K]	$0 \sim 4.5 = 0 \sim 100$
$P_{fluid}$ [kPa]	$2 \sim 30 = 0 \sim 100$
Channel thickness ( $t_{wall}$ )	—

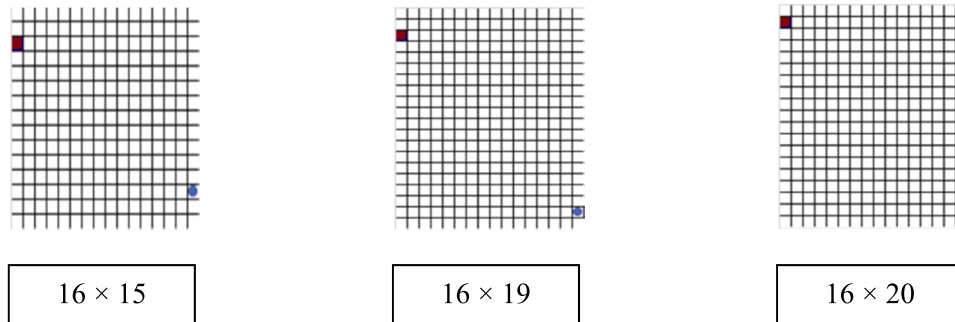


Fig. 3. Reinforcement learning environment.



range. Negative rewards are given whenever an action is performed in the grid environment. and once an agent reaches the outlet, each path will immediately receive a plus reward and start the next simulation. Thus, during the path searching, any additional rewards are not provided to compensation for temperature and pressure. Whenever each path searching is complete, it will receive 4 rewards by the following equation based on temperature and pressure. The reward according to Table 1 is the same as Equation (14).

$$\text{reward} = (\text{goal} - \text{per grid visit count}) + T_{\max} + T_{\text{ave}} + T_{\sigma} + P_{\text{fluid}} \quad (14)$$

### 3.2. CFD analysis environment

In this study, the CFD analysis environment of the cooling channel proceeds with the same flow as show in Fig. 6, and analysis was performed using chtMultiRegionFoam [38,39] of openFOAM. When ANSYS [40] and COMSOL [41], commercial analysis tools, automatically analyze one case through a Python script, it takes an average of 3 h from preprocessing to analysis in the same computer environment, but using openFOAM takes only 20 min on average. We used 4 RTX-3090 GPUs and 2 Intel I-9 CPUs. The total training time of Example 1 was 20.4 days, Example 2 took 24.7 days, and Example 3 took 17.8 days. The above information is added to the 344th line on page 16. Therefore, the modeling of the analysis target was imported using openFOAM, and even the result plot was configured as an automated system. Additionally, in the grid environment, the agent's visits are listed with a Python script to model the cooling channel, and the fluid and solid regions were divided into two, meshed and analyzed.

chtMultiRegionFoam is a solver that analyzes heat transfer between a fluid and a solid in a steady or abnormal state. First, to define the temperature boundary condition of the fluid, the equation for the fluid is analyzed using the temperature of the solid. After that, it is repeated until the method of interpreting the governing equation for the solid using the temperature from the fluid converges. The governing Eqs. (15)–(18) used in this analysis are as follows

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0 \quad (15)$$

Eq. (15) is a continuity equation that expresses the law of conservation of mass, where the first term expresses the mass accumulated or lost in the system over time and the second term expresses the difference between inflow and outflow as flux in the system, where  $\rho$  is the density of the fluid,  $t$  is the time, and  $u$  is the velocity of the fluid.

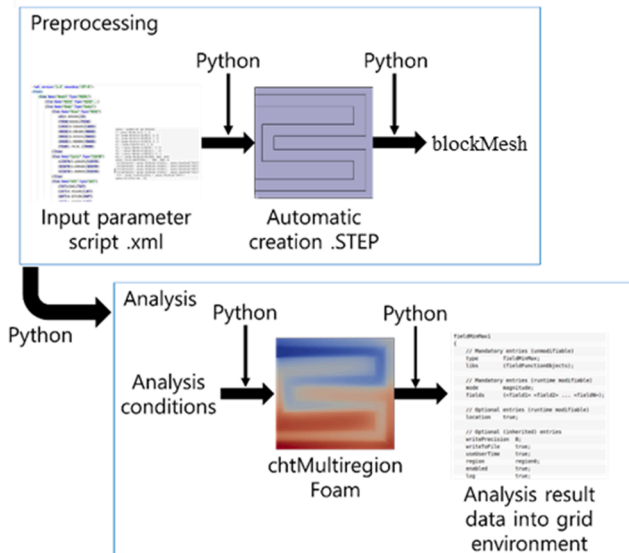


Fig. 6. Flowchart for automatic CFD analysis.

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j) + \rho \epsilon_{ijk} \omega_j u_k = -\frac{\partial p_{\text{rgh}}}{\partial x_i} - \frac{\partial \rho g_j x_j}{\partial x_i} + \frac{\partial}{\partial x_i}(\tau_{ij} + \tau_{tj}) \quad (16)$$

Eq. (16) is the Navier–Stokes equation, where the first term on the left represents the local acceleration of the fluid and the second and third terms represent the acceleration and angular acceleration of convection, respectively, often called convective terms. The first term on the right side represents the surface force (pressure), the second term is the volumetric force (gravity), and the last term is the term for the surface force (shear stress) occurring in the fluid as the viscosity term. Here,  $u$  is the velocity,  $u_r$  is the relative velocity,  $g$  is the gravitational acceleration,  $\tau_{ij}$  is the shear stress, and  $\tau_{tj}$  is the shear stress due to turbulence.

$$\frac{\partial(\rho(k+e))}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j(k+e)) = -\frac{\partial p u_i}{\partial x_j} - \frac{\partial q_j}{\partial x_j} - \rho g_j u_j + \frac{\partial}{\partial x_j}(\tau_{ij} u_i) + \rho r \quad (17)$$

Eq. (17) is the law of conservation of energy and is an expression that shows that the total amount of energy in the system is constant. Energy is neither created nor lost but is converted into other forms of energy. The amount of energy change of a material particle is determined by the energy acting on the particle from the external flux and the energy coming from the internal or mechanical or thermal source. The first term on the left side is the total amount of energy that changes with time, and the second term is the total amount of energy changed by convection. The first term on the right-hand side represents the change in energy due to pressure, the second term is the change in energy due to conduction, the third term is the potential energy of the system, the fourth term is the energy change in viscosity, and the last term is the energy source, where  $k$  is the mechanical energy,  $e$  is the internal energy,  $p$  is the pressure,  $u$  is the velocity,  $\tau_{ij}$  is the shear stress, and  $r$  is the energy source.

$$\frac{\partial(\rho h)}{\partial t} = \frac{\partial}{\partial x_j} \left( \alpha \frac{\partial h}{\partial x_j} \right) \quad (18)$$

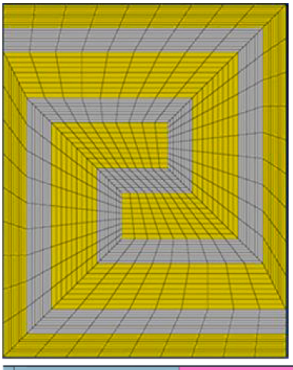
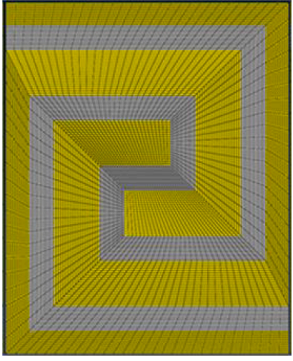
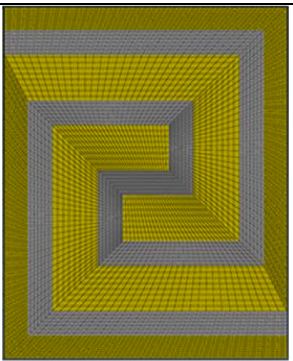
Eq. (18) is a heat transfer equation, and heat transfer analysis was performed on the structural part. The first term on the left represents the change in enthalpy with time, showing that the amount of change in enthalpy is equal to the amount of heat conducted through the solid. Here,  $h$  is the specific enthalpy,  $\rho$  is the density, and  $\alpha$  is the thermal diffusivity. In the coupling of fluid and solid, the temperature at the location where fluid and solid meet was given as isothermal  $T_f + T_s = 0$ . The heat flux also gave the same amount symmetrically as  $Q_f = -Q_s$ , and it can be expressed as  $k_f \frac{dT_f}{dn} = -k_s \frac{dT_s}{dn}$  if the radiant energy condition is neglected.

In CFD analysis, the number and shape of meshes are very important, as well as the skill of the person performing the analysis. The number of surface meshes and interior meshes used in this study are slightly different depending on the shape of the 3D CAD, which was created using the same method as mesh scale 1 in Table 2, and a layer was formed on the wall where the cooling channel and plate meet. To verify the mesh used in the analysis, mesh sensitivity analysis was performed, and the subject was analyzed by selecting one case among the cooling channels created through DQN. The pre-processing program for verification was created using the commercial software ANSA [42], and the analysis conditions have a domain size of 160 mm × 200 mm, a channel thickness of 14 mm, and three mesh scales of 0.5, 1, and 1.5, which were compared.

Table 2 shows the meshes generated in the same way and the analysis results corresponding to scales 0.5, 1, and 1.5. The surface mesh of scale 0.5 was composed of 3,942 pieces, the interior mesh was composed of 18,468 pieces, and the layer was composed of 9 pieces. The surface mesh of scale 1 was composed of 20,650 pieces, the interior mesh was composed of 143,184 pieces, and the layer was composed of 16 pieces. The surface mesh of scale 1.5 was composed of 37,526 pieces, the interior mesh was composed of 383,496 pieces, and the layer consisted of 24 pieces. Even if the number of meshes was increased based on scale 1 used in this study, it was judged that there was little difference in the

**Table 2**

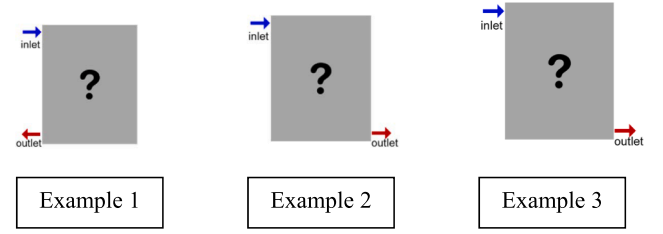
Mesh sensitivity analysis channel results generated by DQN.

	$T_{max}$ : 305.758[K] $T_{ave}$ : 303.329[K] $T_{\sigma}$ : 2.627[K] $P_{fluid}$ : 8.782kPa
Case: 160 × 200_scale 0.5	
	$T_{max}$ : 305.7[K] $T_{ave}$ : 303.286[K] $T_{\sigma}$ : 2.604[K] $P_{fluid}$ : 9.172kPa
Case: 160 × 200_scale 1	
	$T_{max}$ : 305.691[K] $T_{ave}$ : 303.288[K] $T_{\sigma}$ : 2.594[K] $P_{fluid}$ : 9.308kPa
Case: 160 × 200_scale 1.5	

values of  $T_{max}$ ,  $T_{ave}$ ,  $T_{\sigma}$ , and  $P_{fluid}$ . Therefore, the cooling channel reaching the outlet was analyzed based on scale 1.

### 3.3. Example problems

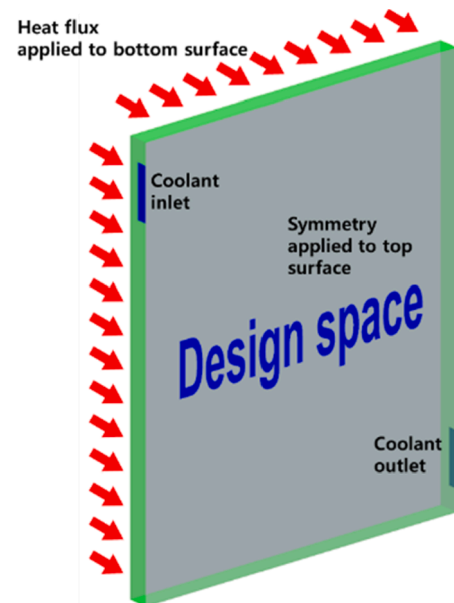
In this study, we aimed to find the shape of a cooling channel with a high reward within the same target variable range by setting the environment of each example differently. Fig. 7 shows three examples when the conditions of the environment are different, and Example 1 and Example 3 were composed based on a 160 mm × 190 mm sample of Example 2. The horizontal width is fixed at 160 mm, and the height is

**Fig. 7.** Conditions of example problems.

150 mm for Example 1, 190 mm for Example 2, and 200 mm for Example 3. Based on the inlet and outlet of Example 2, Example 1 set the outlet direction to the opposite direction of Example 2, and Example 3 set the same direction.

In addition, the openFOAM CFD analysis results were compared and verified with the analysis results of the CFD analysis environment through Ansys. Fig. 8 shows the CFD model for designing the cooling plate. The presented cooling plate model is located between the battery stacks, and heat flux from the battery flows into both sides of the plate. The inlet and outlet of the coolant are placed on the side surface colored in green on the plate, and the analysis using symmetry was performed by modeling only half of the plate.

The CFD problem was defined by referring to the paper of Jarrett and Kim [17], and the conditions for analyzing the cooling plate are shown in Table 3. The material of the cooling plate is aluminium, and the surface is subjected to the no-slip condition. The average temperature chosen in the design procedure is based on the area-weighted average of the heat source. A constant heat flux of 500 W/m<sup>2</sup> was applied as a thermal boundary condition, the initial temperature of the fluid was given as 300 K, and a gauge pressure of 0 Pa was given at the outlet assuming atmospheric pressure [17,18]. In addition, the shape of the fluid part for flow analysis is very narrow at 0.375 mm, and the flow field is divided and composed because the formation of a dense grid is important since there is a parabolic velocity distribution due to the viscous flow. The mesh of the cooling channel was 0.1 mm for body sizing, an across gap of 3 was given to provide a denser mesh shape to the cooling plate and the fluid part, and 0.5 mm was given for the body sizing to the cooling plate. Additionally, in consideration of smooth analysis, analysis time, and improvement of grid quality, it was set as a Tetra mesh.

**Fig. 8.** Cooling channel design space.

**Table 3**  
CFD analysis conditions.

Coolant properties	Value
Coolant fluid	Water-ethylene glycol
Coolant viscosity (PaS)	0.00315 (at 300 K)
Coolant conductivity (W/mK)	0.419
Coolant specific heat (J/kgK)	3494
Coolant density (kg/m <sup>3</sup> )	1065
Plate material	Aluminum
Plate conductivity (W/mK)	871
Plate density (kg/m <sup>3</sup> )	2719
Boundary conditions	
Coolant inlet mass flow (kg/s)	0.001
Coolant inlet temperature (K)	300
Coolant outlet pressure (Pa)	0
Heat flux (W/m <sup>2</sup> )	500

In addition, by applying 16 layers as inflation to the fluid part, the total number of grids in the entire battery flow field was generated on average to be about 2 million.

### 3.4. Build and train deep learning models

The DQN model constructed in this study is the same as shown in Fig. 9 and consists of a main network used for learning and a target network for obtaining actual values. The DQN model uses Python and Keras and consists of five hidden layers and one output layer, and all hidden layers are dense layers.

The three representative types of activation functions are Sigmoid, Tanh, and ReLU. The problem with vanishing gradient is that the input value has little effect on the final layer, because the Sigmoid represents a negative value close to zero. In this study, it was judged that Sigmoid was not suitable because the hidden layer was deep. The second Tanh function is also not suitable because it is a Sigmoid derivative and has the same vanishing slope problem. The reason for using ReLU in this study is that it is the most widely used function and solves the problem of Sigmoid and Tanh gradient annihilation. It is faster to train and less computationally expensive than the above two activation functions.

Numerical experiments were performed to obtain appropriate values for the learning rate, batch size, and number of hidden layers. The learning rate is a hyperparameter that controls how much the model changes in response to the estimated error each time the model weights are updated. Values that are too small can lengthen the learning process and cause stagnation, while values that are too large can learn too quickly or result in an unstable learning process. As the batch size is larger, a wide range of random states can be obtained, and it is an advantage in a problem with a large number of cases like this study. With a small batch size, there are fewer states to replay the experience, so it often only visits the same grid for each episode. The number of hidden layers is theoretically the more the better. If the number of hidden layers is large, mathematically fast convergence is possible and it is not influenced by data noise.

Epsilon decay indicates the degree to which the epsilon value is decreased for each episode. Q-learning can only learn information about

the environment through experience. This is because, if we assume that the epsilon value is the same as the initial value even when we have finished exploring the environment through 1000 episodes, we will have the same policy as if we did not study. The discount factor is expressed as a value between 0 and 1. The closer to zero, the more weight we give to rewards we can get right now than rewards we can get in the future. A value close to 1 indicates that immediate and future rewards are treated equally. Epsilon is the probability of choosing a random action. The agent tries to visit only the grid with the highest Q-value due to the nature of the algorithm. At this time, it is a parameter that gives randomness so that various grids can be explored. The epsilon value was used as 0.3 referring to Ref. [43], and it was also verified that the best result can be obtained when epsilon is 0.3 in this study.

The initial settings of hyperparameters were used by referring to DQN codes implemented by DeepMind [44] and openAI [45]. The DQN code provides guidelines for learning rate, batch size, epsilon decay, discount factor, and initial value of epsilon. In addition, in the original paper, the hyperparameters suitable for the example were selected through numerical experiments. As a result, the hyperparameter values used in this study were determined as learning rate 0.0001, batch size 512, number of hidden layers 5, epsilon decay 0.999, discount factor 0.99 and epsilon 0.3. Mean squared error was used as the loss function, and Adam was used as the optimizer [46,47].

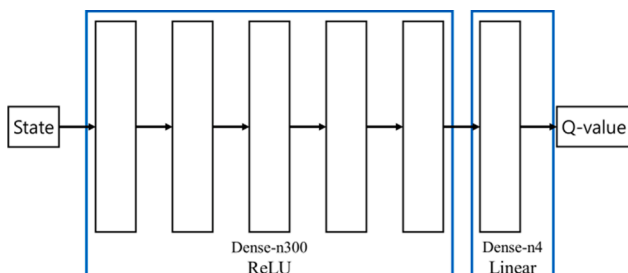
## 4. Results and discussion

In this study, three example problems with specified inlets and outlets were studied, and the results were compared and verified using commercial software. As for the overall domain size, Example 1 was 160 mm × 150 mm, Example 2 was 160 mm × 190 mm, and Example 3 was 160 mm × 200 mm, and in the case of  $t_{wall}$ , the range was 10 mm to 16 mm. All three examples were trained in the same way. Each episode was performed 10,000 times. The target maximum temperature was set as 304.15–306 K, the target average temperature as 302.15–303.5 K, and the target standard deviation as 1–3 K [6–10]. The order of reward importance was set at  $T_{\sigma} > T_{ave} > T_{max} > P_{fluid}$ . Learning ends when the average reward value of the previous 20 episodes exceeds 200, and  $P_{fluid}$  selects a channel with high  $P_{fluid}$  compensation if a channel with  $T_{max}$ ,  $T_{ave}$ , and  $T_{\sigma}$  compensation within 7% is created in the same environment.

Fig. 10(a) shows a graph of the reward obtained through the analysis of the cooling channel created in the grid environment of Example 1, and Example 1 of Table 4 is the result of comparing the CFD analysis results of the channel generated through the DQN algorithm with the Ansys CFD analysis results. Example 1 predicted a total of 1283 cooling channels, and the cooling channel with the highest reward was 340.07. Calculating the error of ANSYS compared to that of openFOAM, the pressure drop was 0.383 kPa, the maximum temperature was 0.728 K, the average temperature was 0.396 K, and the temperature standard deviation was 0.543 K.

Fig. 10(b) shows a graph of the reward obtained through analysis of the cooling channel created in the grid environment of Example 2, and Example 2 of Table 4 is a result of comparing the CFD analysis results of the channel generated through the DQN algorithm with the ANSYS CFD analysis results. Example 2 predicted a total of 894 cooling channels, and the cooling channel with the highest reward was 269.6. Calculating the error of ANSYS compared to that of openFOAM, the pressure drop was 0.678 kPa, the maximum temperature was 0.637 K, the average temperature was 0.589 K, and the temperature standard deviation was 0.608 K.

Fig. 10(c) shows a graph of the reward obtained through the analysis of the cooling channel created in the grid environment of Example 3, and Example 3 of Table 4 is a result of comparing the CFD analysis results of the channel generated through the DQN algorithm with the ANSYS CFD analysis results. Example 3 predicted a total of 615 cooling channels,



**Fig. 9.** DQN architecture.



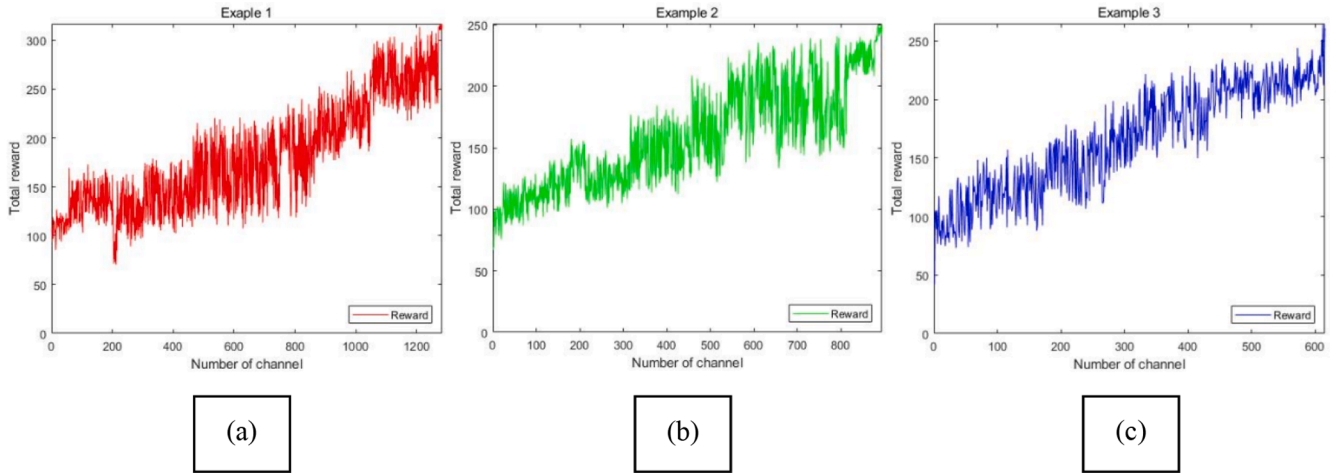


Fig. 10. Each example reward according to the number of channels.

and the cooling channel with the highest reward was 290.19. Calculating the error of ANSYS compared to that of openFOAM, the pressure drop was 0.399 kPa, the maximum temperature was 0.873 K, the average temperature was 0.461 K, and the temperature standard deviation was 0.711 K.

Through the three examples, various cooling channels could be identified within the same target range. In the case of Example 1, when  $t_{wall}$  is 16 mm,  $T_{max}$ ,  $T_{ave}$ , and  $T_{\sigma}$  were close to the minimum values in the set target range, and the best cooling channel shape was predicted in the range of operating conditions. In the case of Example 2,  $T_{max}$  and  $T_{ave}$  except  $T_{\sigma}$  were close to the intermediate values in the target range set when the  $t_{wall}$  was 10 mm. Example 2 shows a good cooling channel, but the result was worse than in Example 1. In the case of Example 3, when the  $t_{wall}$  was 14 mm, the remaining  $T_{max}$  and  $T_{ave}$  except for  $T_{\sigma}$  in the target range set were close to the maximum values, which indicates that Example 3 shows a cooling channel suitable for the conditions of use, but worse than those in the cases of Examples 1 and 2.

In addition, Examples 2 and 3 can be seen to have completely different shapes despite a 10 mm difference in total size. The  $t_{wall}$  of Example 2 had a thickness of 10 mm, and the  $t_{wall}$  of Example 3 had a thickness of 14 mm, whereas the rewards of  $T_{max}$ ,  $T_{ave}$ , and  $T_{\sigma}$  excluding the pressure were 202.95 and 212.77, only a difference of 9.82. Both examples are good cooling channels suitable for the usage conditions in the target range, but a large difference occurred in  $P_{fluid}$ , and Example 3 with a lower  $P_{fluid}$  obtained a higher reward. For further quantitative comparison of ANSYS and openFOAM results, fluid velocity and pressure vector contour results are added in Table 4. It is shown that the average fluid velocity and pressure vector contour between two simulations are exactly the same.

Since the purpose of Q-learning was to search for the shortest path, for the shortest distance condition, negative (–) reward was applied according to the number of grid cells visited. As a result, the agent visited 65 grids in Example 1, 90 grids in Example 2, and 80 grids in Example 3. What is clear here is that depending on the width of the channel, the agent will visit the minimum number of grid cells. Therefore, in order to obtain the shortest distance in the environment of the configured examples, the  $t_{wall}$  should be set to at least 14 mm, and to obtain the minimum value within the target range regardless of the example, the learning episodes should be increased, and the average reward at the end of the learning should also be increased.

Although there is a difference between the analysis results of openFOAM and ANSYS used in this study, since similar errors are shown within a certain range, additional validation of commercial software is not required if an average value is added to the analysis result as a correction value. Additionally, according to the pressure drop compensation, there is a possibility that the maximum thickness will be obtained rather than the minimum fluid thickness that satisfies the target range set by the thickness of the cooling channel. In further studies, it is expected that better results can be obtained if a direct reward is applied to the thickness of the fluid.

## 5. Conclusion

In this study, using the DQN algorithm, a deep learning technique was proposed to predict the cooling channel shape through four variables: target maximum temperature, average temperature, temperature standard deviation, and pressure drop of the battery cooling plate for an electric vehicle.

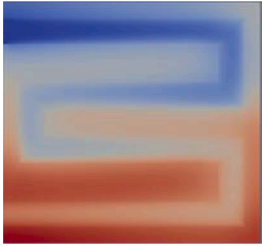
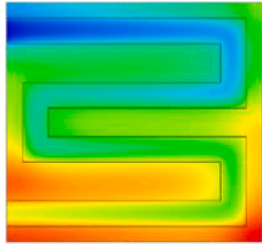

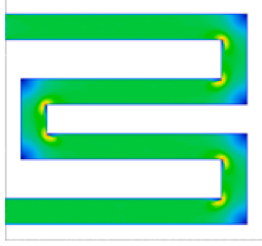
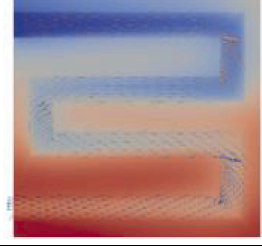
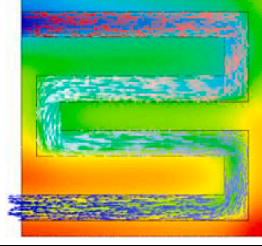

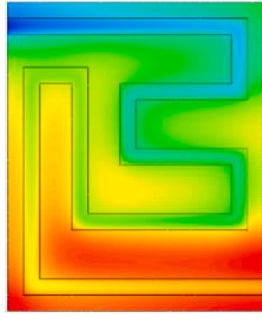


The environment was downscaled for smooth learning of the DQN model, and a result close to the target input was obtained by applying a reward through the simulation result. In addition, it was confirmed that the cooling channel was created in various environments by performing learning on three examples with different inlets and outlets with different overall domain sizes, not one example with the same environment size, inlet, and outlet.

In this study, 10,000 episodes were trained, and results similar to those of commercial software were obtained within a certain level of error by connecting the open-source analysis program and the grid environment. In the proposed method, the analysis is performed on the channel arriving from the inlet to the outlet, and even if the range of target usage conditions is changed, if the inlet and outlet are in the same environment size and location, other cooling channels can be predicted through the learned policy.

As a future study, if the entire inlet and outlet sides of example are designated as ranges in a grid environment, and the number of episodes is increased to train, in this example, the agent finds the optimal inlet and outlet locations through learning, and it is judged that the optimal cooling channel shape can be predicted from the found inlet and outlet locations. Authors believe that cooling channels designed in steady state environments can be used even for the special environments such as



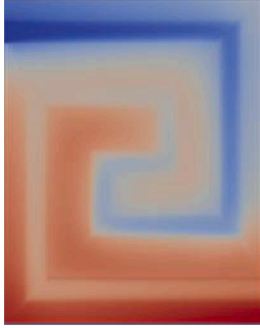
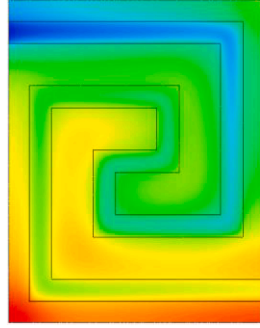

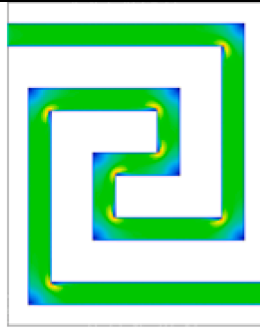
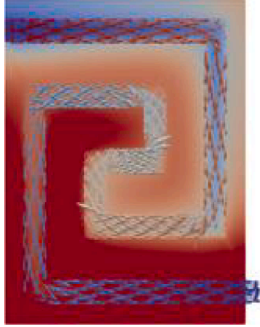
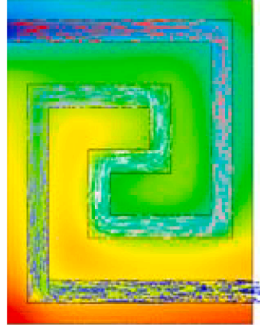


**Table 4**  
Cases result of cooling plate and CFD analysis.

Example	DQN contour	DQN result	ANSYS contour	ANSYS result
1		$t_{wall}$ : 16 mm $P_{fluid}$ : 6.316 kPa $T_{max}$ : 304.203 K $T_{ave}$ : 302.387 K $T_{\sigma}$ : 1.034 K		$t_{wall}$ : 16 mm $P_{fluid}$ : 6.699 kPa $T_{max}$ : 304.931 K $T_{ave}$ : 302.783 K $T_{\sigma}$ : 1.577 K
		Average velocity: 0.156 m/s		Average velocity: 0.156 m/s
		Pressure vector		Pressure vector
2		$t_{wall}$ : 10 mm $P_{fluid}$ : 14.888 kPa $T_{max}$ : 304.928 K $T_{ave}$ : 302.985 K $T_{\sigma}$ : 1.216 K		$t_{wall}$ : 10 mm $P_{fluid}$ : 15.566 kPa $T_{max}$ : 305.565 K $T_{ave}$ : 303.574 K $T_{\sigma}$ : 1.824 K
		Average velocity: 0.25 m/s		Average velocity: 0.25 m/s

(continued on next page)

Table 4 (continued)

		Pressure vector		Pressure vector
3		$t_{wall}$ : 14 mm $P_{fluid}$ : 9.172 kPa $T_{max}$ : 305.7 K $T_{ave}$ : 303.286 K $T_{\sigma}$ : 1.291 K		$t_{wall}$ : 14 mm $P_{fluid}$ : 9.571 kPa $T_{max}$ : 306.573 K $T_{ave}$ : 303.747 K $T_{\sigma}$ : 2.002 K
		Average velocity: 0.179 m/s		Average velocity: 0.179 m/s
		Pressure vector		Pressure vector

uphill or bottleneck traffic conditions by controlling the cooling mass flow rate through the BMS.

This method suggests the possibility of application of deep learning techniques in various fields and is expected to be widely applied to engineering problems through RL.

#### Funding

This study was conducted as a result of research conducted by the Ministry of Trade, Industry and Energy and the Korea Energy Technology Assessment Service (task number 20202020800200, 20192010106780).

#### Declaration of Competing Interest

The authors declare that they have no known competing financial

interests or personal relationships that could have appeared to influence the work reported in this paper.

#### Data availability

No data was used for the research described in the article.

#### References

- [1] S. Panchal, I. Dincer, M. Agelin-Chaab, R. Fraser, M. Fowler, Thermal modeling and validation of temperature distributions in a prismatic lithium-ion battery at different discharge rates and varying boundary conditions, *Appl. Therm. Eng.* 96 (2016) 190–199, <https://doi.org/10.1016/j.applthermaleng.2015.11.019>.
- [2] A.H. Mohammed, R. Esmarelli, H. Aliniagerdroudbari, M. Alhadri, S.R. Hashemi, G. Nadkarni, S. Farhad, Dual-purpose cooling plate for thermal management of prismatic lithium-ion batteries during normal operation and thermal runaway,

- Appl. Therm. Eng. 160 (2019), 114106, <https://doi.org/10.1016/j.applthermaleng.2019.114106>.
- [3] X. Li, F. He, G. Zhang, Q. Huang, D. Zhou, Experiment and simulation for pouch battery with silica cooling plates and copper mesh based air cooling thermal management system, Appl. Therm. Eng. 146 (2019) 866–880, <https://doi.org/10.1016/j.applthermaleng.2018.10.061>.
  - [4] J. Kim, J. Oh, H. Lee, Review on battery thermal management system for electric vehicles, Appl. Therm. Eng. 149 (2019) 192–212, <https://doi.org/10.1016/j.applthermaleng.2018.12.020>.
  - [5] F.F. Bai, M.B. Chen, W.J. Song, Y. Li, Z.P. Feng, Y. Li, Thermal performance of pouch Lithium-ion battery module cooled by phase change materials, Energy Proc. 158 (2019) 3682–3689, <https://doi.org/10.1016/j.egypro.2019.01.891>.
  - [6] P.R. Tete, M.M. Gupta, S.S. Joshi, Developments in battery thermal management systems for electric vehicles: a technical review, J. Energy Storage. 35 (2012), 102255, <https://doi.org/10.1016/j.est.2021.102255>.
  - [7] Z. Zhang, J. Wang, X. Feng, L. Chang, Y. Chen, X. Wang, The solutions to electric vehicle air conditioning systems: a review, Renew. Sustain. Energy Rev. 91 (2018) 443–463, <https://doi.org/10.1016/j.rser.2018.04.005>.
  - [8] J. Yan, Q. Wang, K. Li, J. Sun, Numerical study on the thermal performance of a composite board in battery thermal management system, Appl. Therm. Eng. 106 (2016) 131–140, <https://doi.org/10.1016/j.applthermaleng.2016.05.187>.
  - [9] Z. Li, J. Huang, B.Y. Yann Liaw, V. Metzler, J. Zhang, A review of lithium deposition in lithium-ion and lithium metal secondary batteries, J. power sources. 254 (2014) 168–182, <https://doi.org/10.1016/j.jpowsour.2013.12.099>.
  - [10] Y. Huang, P. Mei, Y. Lu, R. Huang, X. Yu, Z. Chen, A.P. Roskilly, A novel approach for Lithium-ion battery thermal management with streamline shape mini channel cooling plates, Appl. Therm. Eng. 157 (2019), 113623, <https://doi.org/10.1016/j.applthermaleng.2019.04.033>.
  - [11] R. Spotnitz, J. Franklin, Abuse behavior of high-power, lithium-ion cells, J. Power Sources. 113 (1) (2003) 81–100, [https://doi.org/10.1016/S0378-7753\(02\)00488-3](https://doi.org/10.1016/S0378-7753(02)00488-3).
  - [12] H. Teng, Y. Ma, K. Yeow, M. Thelliez, An analysis of a lithium-ion battery system with indirect air cooling and warm-up, S.A.E Int. J. Passenger Cars Mech. Syst. 4 (3) (2011) 1343–1357, <https://doi.org/10.4271/2011-01-2249>.
  - [13] V. Etacheri, R. Marom, R. Elazari, G. Salitra, D. Aurbach, Challenges in the development of advanced Li-ion batteries: a review, Energy Environ. Sci. 4 (9) (2011) 3243–3262, <https://doi.org/10.1039/c1ee01598b>.
  - [14] X. Qian, D. Xuan, X. Zhao, Z. Shi, Heat dissipation optimization of lithium-ion battery pack based on neural networks, Appl. Therm. Eng. 162 (2019), 114289, <https://doi.org/10.1016/j.applthermaleng.2019.114289>.
  - [15] A.K. Thakur, R. Prabakaran, M.R. Elkadeem, S.W. Sharshir, M. Arici, C. Wang, W. Zhao, J. Hwang, R. Saidur, A state of art review and future viewpoint on advance cooling techniques for Lithium-ion battery system of electric vehicles, J. Energy Storage. 32 (2020), 101771, <https://doi.org/10.1016/j.est.2020.101771>.
  - [16] S.H. Yu, S. Sohn, J.H. Nam, C. Kim, Numerical study to examine the performance of multi-pass serpentine flow-fields for cooling plates in polymer electrolyte membrane fuel cells, J. Power Sources. 194 (2) (2009) 697–703, <https://doi.org/10.1016/j.jpowsour.2009.06.025>.
  - [17] A. Jarrett, Y. Kim, Design optimization of electric vehicle battery cooling plates for thermal performance, J. Power Sources. 196 (23) (2011) 10359–10368, <https://doi.org/10.1016/j.jpowsour.2011.06.090>.
  - [18] A. Jarrett, Y. Kim, Influence of operating conditions on the optimum design of electric vehicle battery cooling plates, J. Power Sources. 245 (2014) 644–655, <https://doi.org/10.1016/j.jpowsour.2013.06.114>.
  - [19] S. Panchal, R. Khasow, I. Dincer, M. Agelin-Chaab, R. Fraser, M. Fowler, Thermal design and simulation of mini-channel cold plate for water cooled large sized prismatic lithium-ion battery, Appl. Therm. Eng. 122 (2017) 80–90, <https://doi.org/10.1016/j.applthermaleng.2017.05.010>.
  - [20] G. Calzolari, W. Liu, Deep learning to replace, improve, or aid CFD analysis in built environment applications: a review, Build. Environ. 206 (2021), 108315, <https://doi.org/10.1016/j.buildenv.2021.108315>.
  - [21] G.M. Hobold, A.K. da Silva, Visualization-based nucleate boiling heat flux quantification using machine learning, Int. J. Heat Mass Transfer 134 (2019) 511–520, <https://doi.org/10.1016/j.ijheatmasstransfer.2018.12.170>.
  - [22] B. Kwon, F. Ejaz, L.K. Hwang, Machine learning for heat transfer correlations, Int. Commun. Heat Mass Transfer 116 (2020), 104694, <https://doi.org/10.1016/j.icheatmasstransfer.2020.104694>.
  - [23] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, second ed., MIT Press, Massachusetts Ave, 2018.
  - [24] S. Jónić, T. Janković, V. Gajić, D. Popović, Three machine learning techniques for automatic determination of rules to control locomotion, I.E.E.E. Trans. Bio Med. Eng. 46 (3) (1999) 300–310, <https://doi.org/10.1109/10.748983>.
  - [25] B. Thomas, A. K. John, Machine learning techniques for recommender systems—a comparative case analysis, in: IOP Conference Series, Materials Science and Engineering, IOP Publishing, vol. 1085/1, 2021.
  - [26] S. Sra, S. Nowozin, S.J. Wright, Optimization for Machine Learning, Paperback, MIT Press, Massachusetts Ave, 2011.
  - [27] LECUN, Yann A., et al. Efficient backprop. In: Neural networks: Tricks of the trade. Springer, Berlin, Heidelberg, 2012, pp. 9–48.
  - [28] R. Bellman, Dynamic programming, Science. 153 (3731) (1966) 34–37, <https://doi.org/10.1126/science.153.3731.34>.
  - [29] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fiedland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature. 518 (2015) 529–533, <https://doi.org/10.1038/nature14236>.
  - [30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with Deep Reinforcement Learning. arXiv preprint, 2013.
  - [31] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, AAAI. 30/1 (2016), <https://doi.org/10.1609/aaai.v30i1.10295>.
  - [32] R.A. Howard, Dynamic programming and markov processes, 1960.
  - [33] R. Bellman, A Markovian decision process, Indiana Univ. Math. J. 6 (4) (1957) 679–684, <https://doi.org/10.1512/iumj.1957.6.56038>.
  - [34] M.V. van Otterlo, M. Wiering, Reinforcement learning and markov decision processes, in: Adaptation, Learning, and Optimization, Springer, Berlin, Heidelberg, 2012, pp. 3–42, [https://doi.org/10.1007/978-3-642-27645-3\\_1](https://doi.org/10.1007/978-3-642-27645-3_1).
  - [35] J. Tsitsiklis, B. Van Roy, Analysis of temporal-difference learning with function approximation, Adv. Neural Inf. Process. Syst. 9 (1996), <https://doi.org/10.5555/2998981.2999132>.
  - [36] L.J. Lin, Reinforcement Learning for Robots Using Neural Networks, Carnegie Mellon University, Pennsylvania, 1992.
  - [37] OpenCFD Ltd, OpenFOAM v8 (Version 8), (2000–2021). <<http://www.openfoam.com>>.
  - [38] M. Van Der Tempel, A chtMultiRegionSimpleFoam tutorial, Chamlers University of Technology, Chamlers, 2012.
  - [39] A. Singhal, Tutorial to Set Up a Case for chtMultiRegionFoam in OpenFOAM 2.0.0, University of Luxembourg, Luxembourg, 2014.
  - [40] ANSYS, Engineering analysis system User's manual, 201, 1 & 2, and Theoretical Manual, revision 8.0, Swanson analysis System Inc., Houston, PA, 2021.
  - [41] N. Sivakumar, H. Kanagasabapathy, H.P. Srikanth, Analysis of perforated piezoelectric sandwich smart structure cantilever beam using COMSOL, Mater. Today: Proc. 5 (5) (2018) 12025–12034.
  - [42] ANSA Beta, CAE systems, ANSA 21.0.1, ANSA software manual (ANSA, Rome, Italy, 2020). <<https://www.beta-cae.com/ansa.htm>>.
  - [43] H.W. Kim, W.C. Lee, Real-time path planning for mobile robots using Q-learning, J. IKEEE. 24 (4) (2020) 71–77, <https://doi.org/10.7471/ikee.2020.24.4.991>.
  - [44] DeepMind, Lua/Torch implementation of DQN (DQN 3.0), Andreas Fjeldand, April 07, 2017. <<https://github.com/deepmind/dqn>>.
  - [45] OpenAI, Solving Atari Pong Game w/ Duel Double DQN in Pytorch (OpenAIPong-DQN), MIT, September 19, 2020. <<https://github.com/bhctsntrk/OpenAIPong-DQN/commits/master>>.
  - [46] D. P. Kingma, J. Ba, Adam: a method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980. <<https://doi.org/10.48550/arXiv.1412.6980>>.
  - [47] C. Sammut, G.I. Webb, Encyclopedia of machine learning. Springer Science & Business Media, Eds., 2011.