Proceedings of the ASME 2022
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC/CIE2022
August 14-17, 2022, St. Louis, Missouri

DETC2022-91145

WORK PROCESS TRANSFER REINFORCEMENT LEARNING: FEATURE EXTRACTION AND FINETUNING IN SHIP COLLISION AVOIDANCE

Xinrui Wang
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, USA
xinruiw@usc.edu

Yan Jin*
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, USA
yjin@usc.edu
(*corresponding author)

ABSTRACT

The advancement of artificial intelligence and machine learning technologies has led to significant changes in work processes. The computer agents are applied to perform not only routine and repetitive jobs but also highly complex tasks such as driving a car and steering a ship. Given the sensory information of the environment, a reinforcement learning method has been applied for agents to learn how to perform complex tasks by trial and error through interactions with the environment. To overcome the issues such as limited and sparse training data, researchers are attempting to reuse the previously learned knowledge in new task situations. In this paper, we investigate how feature extraction and finetuning methods can be combined to allow computer agents to perform transfer reinforcement learning more effectively and efficiently in the context of ship collision avoidance. Taking a computer simulation-based empirical approach, we first develop a ship collision avoidance gameplay environment by introducing the own ship, target ships, and the base case and target cases. A deep neural network including four convolutional layers and three fully connected layers is devised for work process feature capturing through deep reinforcement learning. The case study results have shown that features do exist in work processes, and they can be captured and reused. The similarity between the source case and the target case is a key factor that determines how the feature extract and finetuning methods should be combined for effective task results and efficient learning processes.

Keywords: *Deep learning, transfer reinforcement learning, collision avoidance, similarity measures.*

1 INTRODUCTION

Engineering work processes span a wide range of domains, expertise, maturities, and complexities. Some work processes, such as machine processes, are well understood and often hardcoded into machining programs. Other work processes, such as driving a car in a crowded area or steering a ship in congested waterways, can be difficult to prescribe and require human intelligence to complete the associated tasks. As the market demands more and better sophisticated products, robotic applications will increase in both the amount and the level of sophistication. The challenge arises: *how can we train the robots to learn various work processes easily, quickly, and accurately*.

Recent advents in artificial intelligence and machine learning have provided technological means for developing solutions to the problems that used to require the involvement of human beings. Deep learning, among many other techniques, has led to high-quality image recognition, making face recognition a function in today's consumer mobile phones. For learning from past experiences as humans do, reinforcement learning has been adopted for computers to learn to play various games such as Atari [1] and Go [2]. Furthermore, to increase the efficiency of learning and take advantage of the previously learned knowledge, researchers have explored transfer learning to utilize the previously trained neural networks in new task situations [3, 4, 5, 6]. From a perspective of learning work processes, both deep reinforcement learning and transfer learning approaches are fundamental for robots or computer agents to learn and expand their operational knowledge in workplaces. The goal of this research is *to develop mechanisms for work process transfer*

reinforcement learning so that the training for computer agents can be more effective and efficient.

Collision avoidance problems in both vehicle and robotics domains have attracted attention for years due to their practical utility and varying levels of complexity. Kahn et al. proposed and applied an uncertainty-aware model-based learning algorithm to a quadrotor and an RC car obstacle avoidance task [7]. Chen et al. proposed a decentralized multi-agent collision avoidance algorithm based on reinforcement learning, leading to more than 26% improvement in paths quality compared to a state-of-the-art strategy [8]. In this research, the problem of ship collision avoidance is treated as an example of work processes. Our previous work proposed a *belief-based transfer learning* method to apply previously learned neural networks to solve the ship collision avoidance problem in different new encounter situations [8]. After copying the whole network resulting from the previous training cases, the training process in the target situation (i.e., finetuning) is controlled by two parameters, *transfer belief* (i.e., how much the expert's suggested actions should be trusted) and *transfer period* (i.e., how long the expert's network will be consulted). Although this method sports efficiency and simplicity of implementation, its drawback is that the unit of transfer is the whole network, leaving few parameters to adjust to adapt to different new task situations.

To further expand our transfer learning framework, we take a combined *feature extraction and finetuning* approach to transfer reinforcement learning for work processes. More specifically, we seek to address two research questions: 1) *do work processes possess some kind of features*? And if they do, 2) *how can we extract these features and apply them to make the learning process more effective and efficient*? We are rather positive about answering the first question, given that deep learning-based image feature extraction has prevailed for years. We expect similar deep learning neural networks can be applied to extract work features in a reinforcement learning context. For the second question, once the feature extraction can be realized, there will be a large space of variables that can be composed and adjusted to deal with different target task cases. In this research, an empirical approach is taken to investigate the effect of different combinations of the transfer parameters.

The rest of the paper is organized as follows. The related work is reviewed in Section 2, and the details of the methods of this study are described in Section 3. In Section 4, the case study design is described, and the experiment results are presented and discussed. The conclusions are drawn in Section 5, together with future research directions.

2 RELATED WORK

To investigate the learning and transferring of work process knowledge, in this paper, we take a deep reinforcement learning approach and explore various possibilities of retaining and reusing the learned knowledge in ship collision avoidance. Our work is related to deep learning-based feature extraction in image processing, deep reinforcement learning, and transfer learning.

Image feature extraction through deep learning has been highly effective, and significant progress has been made since the early 2010s, thanks to deep learning techniques such as CNN [10]. Nguyen et al. found the off-the-shelf CNN features extracted from general classification training can be successfully transferred to iris recognition [3]. Razavian et al. [4] used features extracted from the Overfeat network [11] to implement different recognition tasks using a different dataset, and surprisingly the performance exceeds high tuned state-of-art method. Singh and Garzon [5] predicted restaurant attributes based on the Yelp images by using VGGNet model [12]. The training and testing accuracy reached 96.2264% and 93.0189%. A Deep transfer learning based model is proposed to predict COVID-19 [13]. The classification of COVID-19 was based on input chest images. ResNet-50 model [13] was used to extract features from input images, and a CNN was used to predict positive or negative results [6].

Deep learning can be used to solve the problem with high-dimensional sensory inputs [1, 14]. From a reinforcement learning perspective, applying deep learning has some challenges. While in supervised learning, the training dataset is usually large and independent, the reinforcement learning dataset is generated dynamically during the learning process and is often sparse, and has delays between action choices and rewards. A sequence of states may also have high correlations. Furthermore, learning new behaviors may change the previous data distribution. Various methods have been proposed to overcome such obstacles [7, 15]. One example is to utilize the experience replay mechanism that can successfully alleviate the training data inefficiency, correlated sequences, and non-stationary data distribution problems [1]. The goal of the learning agent is to maximize the future discounted reward by choosing optimal actions while interacting with the environment.

Transfer learning is a useful technique to solve a new problem based on the experience of a different but related problem. In the deep learning domain, people rarely train an entire model from scratch since it is hard to find a sufficient dataset, and the process is time-consuming. Instead, some pre-trained models based on a very large dataset (such as ImageNet [16, 17]) can be reused or partially reused, which significantly enhances the training efficiency and accuracy. There are two commonly used transfer learning methods in image classification cases [18, 19, 20]. One is *feature extraction*, i.e., for a given target case, find a related base case and copy partial or a whole CNN model pretrained on a large-scale dataset, and then remove the last fully connected layer and add a new one as a classifier based on the new training task on the top of the copied convolutional layers. The other one is *finetuning*, i.e., the process is similar to feature extraction, but some top copied layers or all copied layers are unfrozen for being finetuned through back-propagation during training in the new target case.

In this research, the feature extraction and finetuning methods are taken within a deep reinforcement learning framework. Based on the results of feature extraction, the transfer learning cases are composed of different combinations of the copied-and-frozen layers and copied-and-unfrozen layers

in order to investigate their effectiveness and efficiency. The next section provides details of our methods.

3 METHODS

A computational empirical approach is taken to investigate the issues involved in work process transfer reinforcement learning. Specifically, a gameplay simulation environment of ship collision avoidance is created, and a set of methods are applied to carry out the simulation-based case studies.

3.1 Task environment design

In order to investigate how work processes can be learned based on feature extraction and finetuning, a Pygame based environment of ship collision avoidance was created to conduct case studies, as shown in Figure 1. The white polygon in the center of the dark circle represents the own ship that learns how to avoid the target ships, represented as the green polygons. Only the own ship is trained as a learning agent; it would learn decision-making strategies to reach the goal and avoid collision with target ships. The target ships are assigned with a fixed starting point, destination, and moving speed and direction. The goal of the own ship is to reach the goal which is on the top of the screen.

The white line in Figure 1 represents the current path of the own ship, the length of this line is the exact distance between the goal and the own ship. The large red circle around the own ship represents the expected distance between the own ship and the goal. When the upper end of the white line touches the large circle, and the distance between the own ship and its goal equals the expected distance, it means the own ship successfully reached the goal. The small black circle represents the safe distance of the own ship. If any target ship moves into this circle, it is considered that a collision has happened.

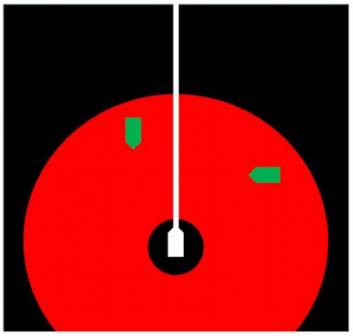

Figure 1: A game environment for case studies

The environment shown in Figure 1 is constructed in a relative coordinate of the own ship. Instead of plotting all the elements in the absolute coordinate as shown in Figure 2(a), the position of the own ship is fixed on the origin, and the direction is fixed to be 90 degrees facing North. The positions of the goal and target ships are set by converting their absolute coordinates to the relative coordinates of the own ship, shown in Figure 2(b). Distances between the own ship and the goal and target ships are denoted as *goal_dist* and *target_dist*, respectively. The position of the goal in the relative coordinate is set by $goal\_dist$ and θ_2. The position of a target ship *t* is set by $target\_dist$ and θ_1. The direction of the target ship is calculated as $\theta_1 + \theta_3 + 90°$.

Because the coordinate system is relative to the own ship, the own ship does not move during the goal-reaching and collision avoidance process. Instead, the goal and target ships will move relatively around the own ship. This relative coordinate system design is close to what is employed in the navigation systems equipped aboard ships.

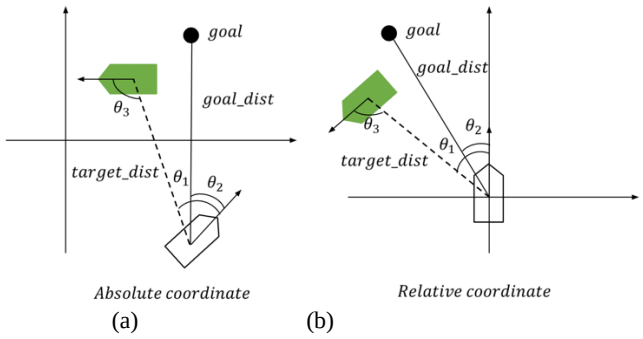

Figure 2: A relative coordinate system

A high-dimensional sensory input deep learning approach [21] is taken, and the pixel values of the game window shown in Figure 1 are treated as the input state. The action choices of the learning agent, i.e., the own ship, are defined in Table 1. Each action is assigned with a pair of linear velocity v (m/s) and angular velocity $\boldsymbol{\omega}$ $(\boldsymbol{rad/s})$ for the agent.

Table 1: Agent Actions

Action	v (m/s)	ω (rad/s)
a_1	7.5	0.1
a_2	12	0.02
a_3	12	0.01
a_4	15	0
a_5	12	-0.01
a_6	12	-0.02
a_7	7.5	-0.1

3.2 Deep reinforcement learning

Based on the gameplay environment described above, a deep reinforcement learning algorithm is devised to allow the own ship to learn from its experiences.

Deep reinforcement learning (Deep RL) is a powerful algorithm to deal with sensory inputs. It utilizes the experience replay mechanism [15], successfully alleviating the training data inefficiency, correlated sequences, and non-stationary data distribution problems. The goal of the learning agent is to maximize the future discounted reward by choosing optimal actions and interacting with the environment. For the ith iteration, the agent chooses an action based on $Bellman\ Equation$ as shown below, where s, a, r, s' denote the current state, action, current reward, and next state, respectively.

$$Q^*(s,a) = \mathbb{E}[r + \gamma * \max_{a'} Q^*(s',a')|s,a)] \quad (1)$$

In practice, a Q-network is used as a function approximator to estimate the above action-value function. A Q-network with weight θ can be trained by minimizing the loss function $L_i(\theta_i)$ at each iteration i,

$$L_i(\theta_i) = \mathbb{E}[(y_i - Q(s,a;\theta_i))^2] \quad (2)$$

where $y_i = \mathbb{E}[r + \gamma * \max_{a'} Q^*(s',a';\theta_{i-1})|s,a)]$ is the target for the ith iteration. The weights are updated by gradient descent. Differentiating the loss function gives the gradient as follows,

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}[(r + \gamma * \max_{a'} Q(s',a';\theta_{i-1}) - Q^*(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)] \quad (3)$$

In this research, a combined model is applied [9]. The training model is constructed by combining Deep RL [1], double Deep Q network [22], and dueling Deep Q network [23]. The double Deep Q network algorithm is able to reduce common overestimations. Dueling Deep Q network architecture separates the Q value into two estimators: the state value and the state-dependent action advantage, which leads to a better approximation of the state values.

3.3 Reward function design

Informative shaping rewards can improve the agent's learning speed by allowing a learning signal to be obtained, even when exploration strategies are simple. [24, 25]. For Deep RL, a composite shaping reward function consisting of piecewise constant and smoothly varying parts is widely used [9, 24, 26]. For vehicle collision avoidance tasks, the ending reward often works as the constant reward. It is obtained when a single episode is ended. A large positive reward is given if the learning agent succeeds, and a large negative reward for failures. The shaping reward is given for every step in the process, based on the task execution. For instance, the distance from the goal or the deviation from the ideal path can be used. It can make the agent explicitly learn more efficient strategies. In this paper, the reward function is designed to be a compound containing sparse components and shaping rewards, as shown below.

$$R_{tot} = w_1 * R_{goal} + w_2 * R_{col} + w_3 * R_{goal-dist} + w_4 * R_{dev} \quad (4)$$

In the above equation, we set $w_1 = 1$, $w_2 = 1$, $w_3 = 0.01$, $w_4 = 0.01$ and they represent the weight of each component. w_3 and w_4 are relatively small because a small negative reward is given for every step of action during the process in order to penalize the agent for staying in the game. $R_{goal} = 200$ and $R_{col} = -200$ are two large sparse rewards that happen only on the final state of each episode: arriving at the goal with 200 rewards or colliding with a target ship with a -200 penalty. The shaping reward $R_{goal-dist}$ is proportional to the distance between the own ship and the goal (i.e., $goal\_dist$ in Figure 2), reflecting the moving process of the own ship. R_{dev} relates to the deviation angle between the own ship's moving direction and the shortest path to the goal (i.e., θ_2 in Figure 2); it is designed to force the agent to avoid obvious deviations from the shortest path to the goal. The shortest path is simply a line connecting the current position of the own ship and the goal.

The shaping rewards are defined as shown below.

$$R_{goal-dist} = k_1 \times goal\_dist \quad (5)$$

$$R_{dev} = k_2 \times \theta_2 \quad (6)$$

where $k_1 = -1$, $k_2 = -5$. k_1 is set to be smaller than k_2. Because $goal\_dist$ is large before the own ship gets closer to the goal, θ_2 is small due to low angular velocity. The penalty for these two components should be balanced. In this way, the agent receives a small penalty if it is far from the goal or makes deviations, forcing it to move closer to the goal and converge to the shortest path (or almost shortest path) sooner than being trained without shaping components. All the parameters are determined based on the experiments, multiple settings are tried, this combination gave the best training results, and the cumulated reward converged more quickly and stable compared to others. Reward function design is crucial to training; more research will be done in the future to investigate the effect of different shaping and parameters.

3.4 Measuring similarity

To realize transfer reinforcement learning in work process applications, an agent needs to learn from a source task situation and transfer the learned knowledge to the target task situations. One important relationship between the source task and the target task is their level of *similarity*. Typically, the *dissimilarity* measure is defined as weighted normalized Euclidian distance between two points in a specific task complexity space [27, 28], as shown in Eq. (4),

$$Dist_{a,b} = \sum_{i=1}^{n} w_i \times (x_{ai} - x_{bi}) \quad (7)$$

where $Dist_{a,b}$ represents the distance between $case\ a$ and $case\ b$. x_{ai} and x_{bi} are the i^{th} characteristic of $case\ a$ and $case\ b$, respectively, and w_i are the corresponding weights. There are also other vector distance measures being used to represent similarity [29, 30]. Autonomous ship collision avoidance scenarios are often complicated, involving different ideal paths of the own ship, different number of target ships, as well as their different destinations and directions. Thus, the base case and target case may contain different numbers of components, making it hard to use distance calculation to capture the similarity between the two cases. On the other hand, when all the attributes in the two cases can be categorized into *common* ones and *different* ones, the following definition can be taken to define similarity measures [31].

$$SIM(a,b) = \frac{\alpha \times common}{\alpha \times common + \beta \times different} \quad (8)$$

where $common$ represents the number of common attributes between $case\ a$ and $case\ b$, $different$ represents the number of different or missing attributes between $case\ a$ and $case\ b$. α and β are corresponding weights of common attributes and different attributes. In this research, an adjusted similarity measuring method based on the above ratio model is applied as described in Section 4.

3.5 Feature extraction and finetuning

The goal of this research is to explore whether ship collision avoidance as a work process possesses some kind of features and, if these features exist, to assess whether and how the features can be transferred to similar or dissimilar tasks. For achieving this goal, a deep reinforcement learning neural network is devised for transfer reinforcement learning study, as shown in Figure 3.

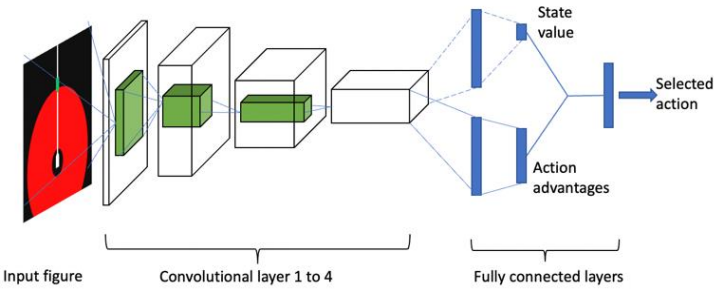

Figure 3: Structure of transfer reinforcement learning

Source network vs. target network: From a transfer learning perspective, the neural network resulting from training in the base case is called the *source network*, and the *target network* is to be developed, through either *transferring* or *training* or both, for the target task cases. In this study, the source network and target network share the same structure, which is constructed by four convolutional layers and fully connected layers, as shown in Figure 3. The first convolutional layer convolves 32 8 × 8 filters with stride 4. The second convolutional layer convolves 64 4 × 4 filters with stride 2. The third convolutional layer convolves 64 3 × 3 filters with stride 1. The fourth convolutional layer convolves 200 7 × 7 filters with stride 1. The following fully connected layers are designed with a dueling structure. For source networks, features are learned and stored in the *convolutional layers*. For target networks, convolutional layers are either *copied* from the source network or *randomly initialized* depending on different transfer strategies.

For both the source network and target network, the *fully connected layers*, as shown in Figure 3, work for decision-making in RL cases in similar ways as the fully connected layers working as a classifier to predict the label of input figures in supervised learning. Instead of predicting labels, actions are selected from the predefined action space based on input figures.

Feature extraction: Feature extraction is a method to transfer learned knowledge (i.e., *convolutional layers* of the source network) resulting from training with the base case to a target case, e.g., coping partial or the whole convolutional layers. In addition, one also needs to remove the fully connected layers of the source network and add new ones in the target network as a classifier, trained for the new task, on the top of the copied convolutional layers. The copied convolutional layers are treated as a *fixed feature extractor* and do not need to be trained again because they have already contained useful features learned from the base dataset. In this study, we use the feature extractor to extract the meaningful features in the base dataset and train the rest of the model to complete the new task cases.

Finetuning: Finetuning is a method similar to feature extraction. The difference is that instead of freezing all the copied convolutional layers, some higher-level layers (i.e., the right-hand side convolutional layers in Figure 3) or all copied layers are unfrozen for being finetuned by backpropagation during training in the target task case. When the similarity between the base case and target cases is low, it is necessary to use the finetuning method. The copied convolutional layers may contain some dataset-specific features because of the big difference between the two datasets; thus, it might work better to be updated together with added randomly normalized fully connected layers. One should be cautious about using finetuning method when the new training dataset is small; it may lead to overfitting issues [16].

If the work process does have features, it will be captured and stored in convolutional layers of the source network. If the extracted features, as well as decision-making strategies in fully connected layers, can be copied and reused in the target network in a proper way, the learning efficiency and accuracy can be enhanced.

4 CASE STUDY

With the methods described above, a set of experiments have been conducted. In this section, we first illustrate the three pairs of transfer learning cases and then present the results and the ensuing discussions.

4.1 Base cases and target cases

In Figure 4, each row represents a pair of a base case (on the left) and a target case (on the right). There are three rows, hence three pairs of cases. The target cases are created by adding a target ship to base cases. Thus, target cases are more complex than base cases. Assuming the own ship and target ships have the same speed, for $case3$ and $case4$, if the own ship doesn't change its direction, collision will happen in the center of the screen between the own ship and the common target ship in the two cases. The same thing happens for $case5$ and $case6$; the collision between the own ship and upper target ship will happen if keeping the initial direction. Thus, only the ideal (shortest) path of the own ship in $case1$ and $case2$ is the direct straight line connecting the starting point of own ship and its goal on the top of the screen. The ideal paths in other cases are curves to avoid collision with target ships. These curves are different from each other based on the target ship settings.

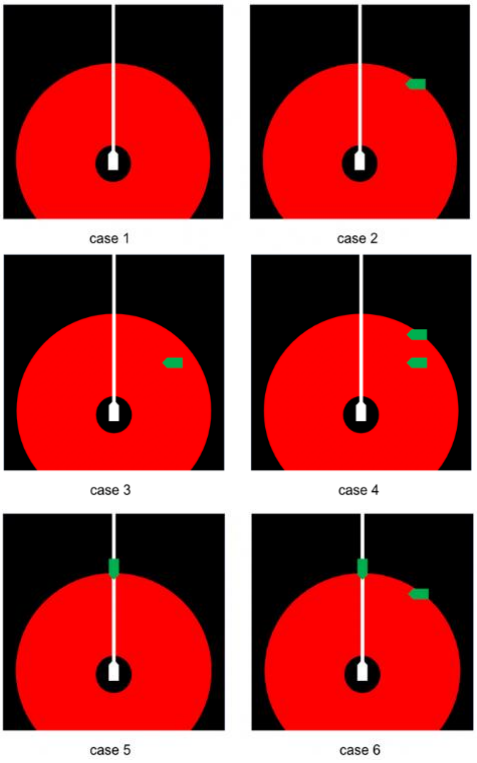

Figure 4: Base cases and target cases

The similarity measures between base cases and target cases are defined following Eqn (8). Since we aim to measure the similarity between two cases, more weight should be given to the common category. For this reason, α should be larger than β. In this research we set $\alpha = 1,\ \beta = 0.5$. Although for general ratio-based similarity measures, the value of $common$ and $different$ represents the number of same and different components in between two cases, in this research, the effect of components differs. For example, the white line representing the path between the own ship and the goal takes more space and looks more obvious than the target ships; it should take more weight intuitively. And according to experiment results, transferring the source network pretrained by images containing only the path can make the target network converge sooner and achieve a higher reward than transferring the source network pretrained by images only containing the target ship based on the same target case. The path component should be assigned more weight than the target ship component since it plays a guiding role in the ship driving scenario. What is more, when adding an extra target ship, even if the target ship is less important, its direction may affect the direction of the path. It is more likely for the own ship to change the path if the added target ship does not follow the same direction as the existing target ship. More weight should be assigned to the target ship with a new direction than following the existing direction when calculating $different$. For these reasons, $common$ and $different$ calculation in similarity measures should not simply be the number of common and different components. The corresponding weights should be taken into consideration. The modification of Eqn (8) follows the rules below:

- $common = 0, different = 0$ initially, since no attributes are categorized yet. Then these two categories will be added with different value when comparing all attributes one by one.
- If the base case and target case have the same ideal path (like in $case1$ and $case2$), $common \leftarrow common + 1$, otherwise $different \leftarrow different + 1$.
- If the base case and target case contain the same target ship, $common \leftarrow common + 0.5$
- If the target case has an extra target ship than the base case, the direction of this target ship exists in the base case, $different \leftarrow different + 0.2$. If the direction of the extra target ship doesn't exist in the base case, $different \leftarrow different + 0.4$.

The value of α and β, as well as the weights of different components, are assigned due to the above considerations and multiple trials. Based on the modified formula, similarities between 3 pairs of cases are calculated as shown in Table 2. Since similarity is sensitive to the parameter settings, more investigation should be done in the future to figure out the relationship between these parameters and the resulting similarity, deriving the more accurate expression for similarity measures.

Table 2: Similarity between base cases and target cases

Base case	Target case	Similarity
$\boldsymbol{case1}$	$case2$	0.83
$\boldsymbol{case3}$	$case4$	0.45
$\boldsymbol{case5}$	$case6$	0.42

Both feature extraction and finetuning methods are employed in studying transfer reinforcement learning in ship collision avoidance work processes. Details of realization in 3 pairs of transfer learning cases are as follows:

- *Feature extraction*: Pretrain the source network in the base cases ($case1, case3, case5$), copy the weight of n ($1 \leq n \leq 4$) convolutional layers to the target network and freeze them. Randomly initialize and train other layers in corresponding target cases ($case2, case4, case6$).
- *Finetuning*: Copy n ($1 \leq n \leq 4$) layers or the whole network, including fully connected layers (for feature extraction, copying and freezing the whole model is meaningless) to the target network and unfrozen the copied layers. Randomly initialize the rest and update the whole target network in the target cases.

4.2 Results and discussion

Feature extraction and finetuning methods were applied on three pairs of base case and target case; the similarity of each pair changes from high to low, as indicated in Table 2. In order to verify the existence of features in the ship's goal-reaching and collision avoidance working process, the transfer reinforcement learning, or TRL for short, training results are compared with three *bootstrap* target cases training results which are trained from scratch (randomly initialize the whole network without

transferring weights from the source network, explore the environment following ε-greedy policy). In order to investigate the transferability of features through each layer, different layer transfer from the bottom (closer to the input images) to the top (closer to the output actions) is applied. For example, "2layers" shown in the plotting means transferring the parameters of the first and second layers on the bottom from the source network to the target network. To control the variable, exploration probability ε is set to be 1 at the beginning and 0.0001 at the end for both learning from scratch and TRL approaches. Training results are shown in Figures 5, 6, and 7. The horizontal axis in the figures represents the number of training episodes, and the vertical axis represents the total reward for each episode. Each result is obtained by running with 15 different random seeds; the average reward is represented by the solid line, and the variation is shown with the shaded area.

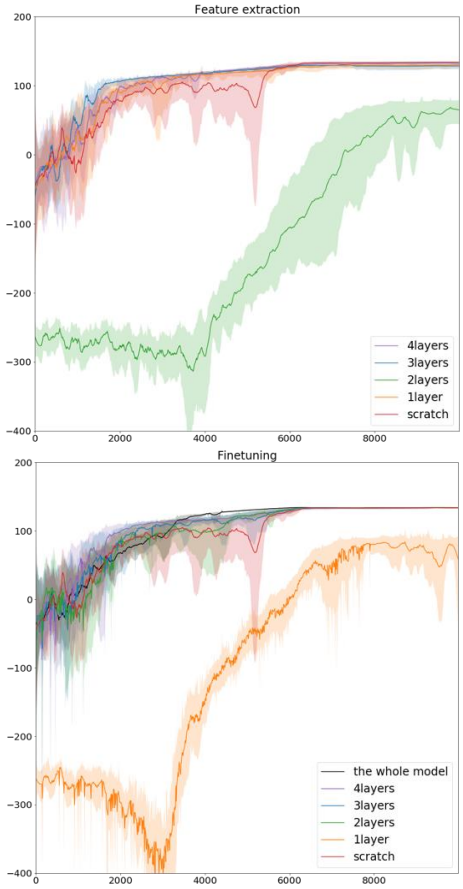

Figure 5: Case 2 training results with TRL from Case 1

High-level similarity cases (case1 to case 2): As shown in Figure 5, when the similarity between the base case and target case is high, both TRL approaches, on average, perform better than learning from scratch except for two specific layers. The average reward converges earlier, and the variation is less, which implies the training is more stable. The comparison shows that even though the target case is more complex than the base case, there still exist some common features in both cases. The features were extracted and stored in the source network when it was being trained in $case\ 1$. The target network learned these features by copying the parameters from the source network, accelerating the convergence in $case\ 2$. And also, because of the high similarity between the two cases, decision-making strategies in $case\ 1$are also useful for $case\ 2$. Finetuning the copied whole model, including the fully connected layers method, allows the decision-making strategies to be transferred to the target network, leading to better performance than all other transfers.

Feature extraction makes the average reward higher and converges earlier compared to the finetuning approach. It indicates that the copied layers contain most of the transferred features, and not much updating is needed for high similarity cases. For finetuning method, except for 1-layer finetuning, the average reward and converging rate increase with the number of transferred layers. It implies features were stored in all four layers. More useful features will be transferred to the target network if more layers are transferred from the source network. The average reward of 3-layers feature extraction is higher in the process and converges earlier than 4-layers feature extraction when utilizing the feature extraction method, and it is also more stable. This phenomenon shows us the fourth convolutional layer is more specific to $case\ 1$. The copied features of $case\ 1$ in the fourth layer do not work well for $case\ 2$ due to the difference between the two cases. Finetuning the fourth layer to make it more specific to $case\ 2$ is needed. So, copying and freezing all convolutional layers may not be a good idea, even when similarity is high.

Regarding the undesired performance of 2-layer feature extraction and 1-layer finetuning, it may be caused by a fragile co-adapted structure on successive layers because the drop can be fixed by adding or removing one copied layer. For 2-layer feature extraction, two layers on the bottom were copied from the source network and kept frozen, and the third layer was randomly initialized and updated. As a result, the features cannot be relearned by updating the third layer alone because the second and the third layers are highly co-adapted, and features in these two layers interact in a complex way. Finetuning is needed due to this co-adaption issue. The 2-layer finetuning achieves optimal performance because the second and third layers were jointly trained.

For 1-layer finetuning, the reason for the sharp drop of performance is the highly co-adapted structure between the first and second layers on the bottom. The first layer was transferred from the source task, and the second layer, as well as the rest of the model, were randomly initialized. As we know, the first layer was well-trained in the base case; it does not need to be updated a lot, although in this case, the transferred first layer was jointly trained with the second layer, which was randomly initialized and needed to be updated. Features in these two layers may interact in a particular way, which cannot be relearned if the training process starts with one well-trained layer and another random layer, making the performance worse than learning from scratch. Freezing the well-trained layers can solve this kind of co-adaption, as shown in the 1-layer feature extraction method.

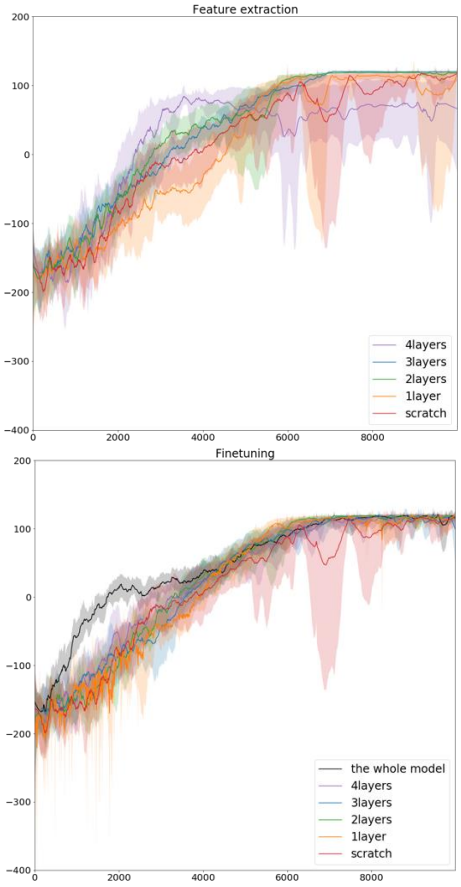

Figure 6: Case 4 training results with TRL from Case 3

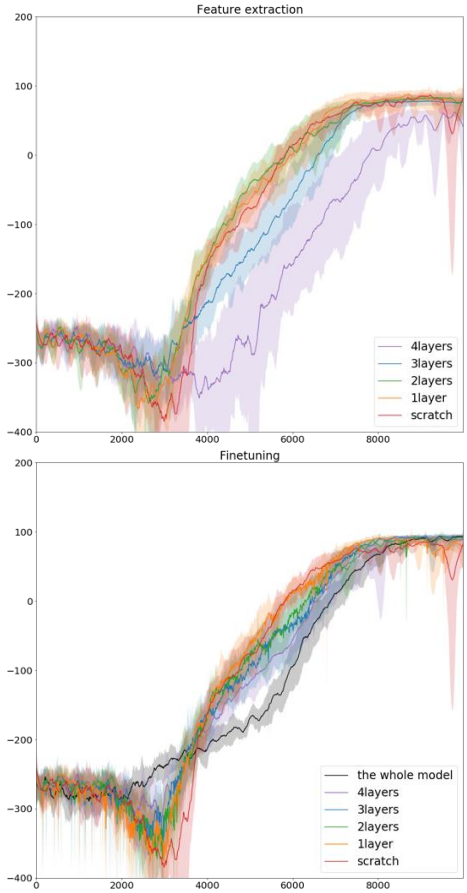

Figure 7: Case 6 training results with TRL from Case 5

Low-level similarity cases (case 3 to case 4): For *case* 3 and *case* 4, the similarity becomes lower. TRL approaches still enhance the learning efficiency compared to learning from scratch, except for 4-layer feature extraction. The sharp drop caused by co-adaption disappears. The finetuning method has a higher and more stable average reward compared to the feature extraction method. Due to the lower similarity, finetuning is needed for the copied layers to become more specific to the target case.

For the feature extraction method, 4-layer feature extraction has the worst performance because all the convolutional layers were frozen, and the difference between the base and target case was fixed. For finetuning method, the whole model finetuning has the highest reward at the beginning because the decision-making layer contains collision avoidance strategies is transferred. As introduced in section 4.1, a collision with a target ship will happen if the own ship keeps the original direction. Transferring the decision-making layer as well as convolutional layers containing collision avoidance features helped the agent avoid the penalty of hitting target ships, and 1-layer finetuning converged earliest instead of the whole model finetuning. General features were learned by transferring the first layer, and there was no need to fix the difference in other layers, so 1-layer finetuning has the highest convergence rate.

Lowest-level similarity cases (case 5 to case 6): For the lowest similarity pair *case* 5 and *case* 6, the benefits of TRL still exist for some low-level transfer. Finetuning methods converge to a higher value compared to the feature extraction method; like in the pair of *case* 3 and *case* 4, differences between the base case and target case need to be fixed.

Unlike the higher similarity cases, where the average reward increases with the number of transferred layers, the average reward decreases when more layers are transferred. For the feature extraction method, 4-layer feature extraction works even worse than learning from scratch since the differences in frozen layers are not able to be fixed. For the finetuning method, there is a drop at around 3000 episodes caused by the penalty of collision except for the whole model finetuning. Similar to the pair of *case* 3 and *case* 4, the whole model finetuning method transfers collision avoidance strategies in the decision-making layer, allowing the agent to evade the collision penalties during the early episodes. Because the agent cannot experience and learn from the early penalty phase, it loses the chance for a sharp reward increasing as the agent in other cases did after the penalty.

Target cases were retained in the case base and can be further used as base cases. The capability of the system was improved by the extension of stored experiences.

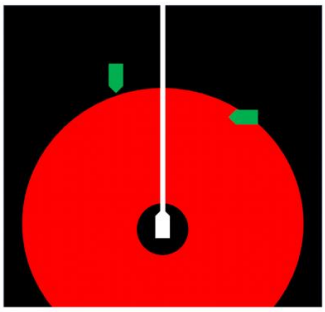

Figure 8: A new collision avoidance case

As shown in Figure 8, given a new task that is different from all cases, cases 1 to 6 can be selected as the base case based on similarity. The similarity between *case* 2 and the *new case* is higher than others, *case* 2 is considered to be the base case. The network trained by the whole model finetuning method has the best performance; it can be used as a representative case solution in *case* 2. For solving the given new task, the representative network in *case* 2 is used as the source network.

For previous cases, the existence and transferability of features were investigated by comparing to learning from scratch that follows a ε-greedy policy (ε drops from 1 to 0.001). The training for TRL also uses the same ε value. For this *new case*, the purpose is to test the reusability of the representative source network from *case 2*. Therefore, ε can be set smaller at the beginning to shorten the exploration period and decrease the randomness when applying TRL. Since the similarity between *case 2* and the *new case* is high, ε is set to be 0.1. As shown in Figure 9, the average reward of the reused network converges much earlier than learning from scratch and converges to the same value as learning from scratch. It indicates that the reuse of stored cases is applicable and efficient.

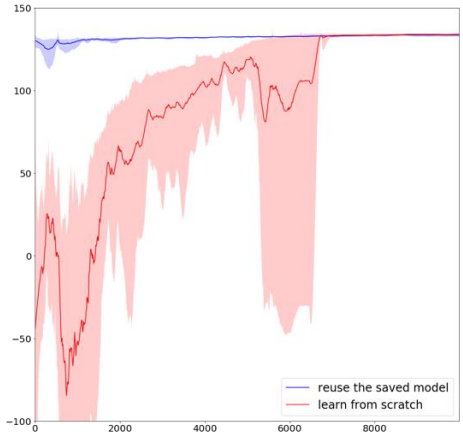

Figure 9: Reuse the saved finetuned model

5 CONCLUSIONS AND FUTURE WORK

Computer agents, including vehicles and robots, can learn complex work processes through reinforcement learning. In order to increase the learning effectiveness and efficiency, the agents need to employ a transfer reinforcement learning methodology in which they reuse the previously learned neural networks and adapt them to the new task situations through much more efficient training processes. In this paper, a computer simulation-based empirical approach is taken to investigate how the feature extraction and finetuning methods can be properly combined to achieve successful transfer reinforcement learning in the context of ship collision avoidance. Through the findings revealed from the results of the experiments, the following conclusions can be drawn.

1. There are *features* in the work process, and they can be captured and reused by transfer reinforcement learning.
2. When the *similarity* between the base case and target case is high, the feature extraction method is more efficient, and the average reward increases with the number of transferred layers. Conversely, in low similarity cases, the average reward decreases when more layers are transferred, and applying the finetuning method becomes more important.
3. For the feature extraction method, at least the last convolutional layer should be unfrozen and trained with the target case for avoiding slow convergence and inferior results.
4. High-level feature co-adaption in successive layers happens when the similarity is high, causing the performance to drop when separating the layers. Introducing more convolutional layers may alleviate this issue.
5. Transferring fully connected layers allows agents to speed up early learning but deprives their chances to learn from making mistakes.

It is worth mentioning that the findings and conclusions described above are limited to the ship collision avoidance type of work processes and the level of complexity of testing cases. Our ongoing work expands the testing cases into more realistic collision avoidance situations and will also consider the work processes with heterogeneous agents. Furthermore, both the feature extraction and the co-adaption can be sensitive to the similarity measure. We plan to explore different similarity measures in future studies.

6 ACKNOWLEDGEMENTS

This paper is based on the work supported by the Autonomous Ship Consortium (ASC) with members of BEMAC Corporation, ClassNK, MTI Co. Ltd., Nihon Shipyard Co. Ltd. (NSY), Tokyo KEIKI Inc., and National Maritime Research Institute of Japan. The authors are grateful for their support and collaboration on this research.

7 REFERENCES

[1] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. “Playing atari with deep reinforcement learning.” arXiv preprint arXiv:1312.5602 (2013).

[2] Greenemeier, Larry. “AI versus AI: Self-Taught AlphaGo Zero Vanquishes Its Predecessor”. Scientific American. Retrieved 20 October 2017.
[3] Nguyen, K., Fookes, C., Ross, A., & Sridharan, S. (2017). Iris recognition with off-the-shelf CNN features: A deep learning perspective. IEEE Access, 6, 18848-18855.
[4] Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 806-813).
[5] Singh, D., & Garzon, P. (2015). Using CNN and transfer learning to perform yelp restaurant photo classification. http: //cs231n.stanford.edu/reports/2016/pdfs/001_Report.pdf
[6] Pathak, Y., Shukla, P. K., Tiwari, A., Stalin, S., & Singh, S. (2020). Deep transfer learning based classification model for COVID-19 disease. Irbm.
[7] Kahn, G., Villaflor, A., Pong, V., Abbeel, P., & Levine, S. (2017). Uncertainty-aware reinforcement learning for collision avoidance. arXiv preprint arXiv:1702.01182.
[8] Chen, Y. F., Liu, M., Everett, M., & How, J. P. (2017, May). Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In 2017 IEEE international conference on robotics and automation (ICRA) (pp. 285-292). IEEE.
[9] Liu, X., & Jin, Y. (2020). Reinforcement learning-based collision avoidance: impact of reward function and knowledge transfer. AI EDAM, 34(2), 207-222.
[10] Manjunath Jogin; Mohana; M S Madhulika; G D Divya; R K Meghana; S Apoorva (2018), Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning, 3rd IEEE Int’l Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT).
[11] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.
[12] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
[13] Rezende, E., Ruppert, G., Carvalho, T., Ramos, F., & De Geus, P. (2017, December). Malicious software classification using transfer learning of resnet-50 deep neural network. In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1011-1014). IEEE.
[14] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.
[15] Lin, L. J. (1992). Reinforcement learning for robots using neural networks. Carnegie Mellon University.
[16] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). IEEE
[17] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2
[18] Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 806-813).
[19] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014, January). Decaf: A deep convolutional activation feature for generic visual recognition. In International conference on machine learning (pp. 647-655). PMLR.
[20] Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. Advances in neural information processing systems, 27.
[21] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.
[22] Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 30, No. 1).
[23] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June). Dueling network architectures for deep reinforcement learning. In International conference on machine learning (pp. 1995-2003). PMLR.
[24] Popov, I., Heess, N., Lillicrap, T., Hafner, R., Barth-Maron, G., Vecerik, M., ... & Riedmiller, M. (2017). Data-efficient deep reinforcement learning for dexterous manipulation. arXiv preprint arXiv:1704.03073.
[25] Badnava, B., & Mozayani, N. (2019). A new potential-based reward shaping for reinforcement learning agent. arXiv preprint arXiv:1902.06239.
[26] Ji, H., & Jin, Y. (2021). Evaluating the learning and performance characteristics of self-organizing systems with different task features. AI EDAM, 1-19.
[27] Changchien, S. W., & Lin, M. C. (2005). Design and implementation of a case-based reasoning system for marketing plans. Expert systems with applications, 28(1), 43-53.
[28] Liao, T. W., Zhang, Z., & Mount, C. R. (1998). Similarity measures for retrieval in case-based reasoning systems. Applied Artificial Intelligence, 12(4), 267-288.
[29] Tadrat, J., Boonjing, V., & Pattaraintakorn, P. (2012). A new similarity measure in formal concept analysis for case-based reasoning. Expert Systems with Applications, 39(1), 967-972.
[30] Duverlie, P., & Castelain, J. M. (1999). Cost estimation during design step: parametric method versus case based reasoning method. The international journal of advanced manufacturing technology, 15(12), 895-906.
[31] Tversky, A. (1977). Features of similarity. Psychological review, 84(4), 327.

