Proceedings of the ASME 2018 International Design Engineering Technical Conference & Computers and Information in Engineering Conference
IDETC/CIE 2018
August 26-29, 2018, Quebec City, Quebec, Canada

DETC2018-86013

DESIGN OF TRANSFER REINFORCEMENT LEARNING UNDER LOW TASK SIMILARITY

Xiongqing Liu
IMPACT Laboratory
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, California 90089
xiongqil@usc.edu

Yan Jin*
IMPACT Laboratory
Dept. of Aerospace & Mechanical Engineering
University of Southern California
Los Angeles, California 90089
yjin@usc.edu
(*corresponding author)

ABSTRACT

In this paper, a deep reinforcement learning approach was implemented to achieve autonomous collision avoidance. A transfer reinforcement learning approach (TRL) was proposed by introducing two concepts: transfer belief – how much confidence the agent puts in the expert's experience, and transfer period – how long the agent's decision is influenced by the expert's experience. Various case studies have been conducted on transfer from a simple task – single static obstacle, to a complex task – multiple dynamic obstacles. It is found that if two tasks have low similarity, it is better to decrease initial transfer belief and keep a relatively longer transfer period, in order to reduce negative transfer and boost learning. Student agent's learning variance grows significantly if using too short transfer period.

Keywords: Deep reinforcement learning, transfer learning, collision avoidance, task complexity, inter-task similarity

1 INTRODUCTION

Collision avoidance has been widely studied and researched in many industrial fields over the years, and is a crucial component for building self-driving systems. In the area of robotics, research has been focused on issues related to how vehicle robots avoid obstacles as well as each other (Brunn, 1996; Alonso-Mora, 2013; Shiomi et al, 2014) and how assembly robots, or manipulators, avoid interferences among its own arms or with those of others (Hourtash et al, 2016; Hameed & Hasan, 2014). Another major field where collision avoidance represents a major problem is transportation. Self-driving cars must be able to avoid obstacles and other vehicles in various situations (Mukhtar et al, 2015). In the shipping industry, collision avoidance can be highly difficult, when the water areas are becoming congested, due to the large inertia of ships causing immovability when movement is needed (Goerlandt & Kujala, 2014). Once a collision happens at sea, the loss can be tremendous (Eleftheria et al, 2016). Airplane collision avoidance (Zou, 2016) and even the collision with debris in space (Casanova et al, 2014) have become issues due to the increasing level of congestion.

Different approaches have been proposed to solve collision avoidance problems, which can be divided into two large categories, one is vehicle control system development and the other traffic system development. Vehicle control can be further categorized into the dynamical systems approach (e.g., Machado et al, 2016), which relies on traditional control theories, and the intelligent systems approach (e.g., Yang et al, 2017), which applies knowledge systems and machine learning techniques. While the dynamical systems approach can be effectively applied in mostly predictable circumstances, when the uncertainty level becomes high and exceptions happen, the intelligent systems approach will be needed. Traditional knowledge based systems have been applied to collision avoidance (Jin and Koyama, 1987). However, the issues of knowledge acquisition, formalization and management have remained to be practically challenging.

Traditional collision avoidance algorithms will suffer when the environment can be quite dynamic, with random obstacle and other vehicles moving at a random speed. It is necessary to build a system that can efficiently learn from its own mistake and past experience. The recent progress in machine learning, especially deep learning (LeCun et al, 2015), has opened the ways to

developing systems that can learn from humans' operation experiences (e.g., through supervised deep learning) and from machines' own experiences (e.g., through reinforcement learning). The reinforcement learning approach allows an agent to learn from its past experience. By interacting with the environment, the agent learns to select actions at any state to maximize the total reward. In case of deep learning, e.g., AlphaGo (Chen, 2016), the agent learns from the experience of human experts and apply the learned skills to solving the problems in the same domain of the experts.

One common observation about the current deep learning systems, including AlphaGo, is that they can only function well within the narrow domain of the tasks that they are trained to work for. This observation manifests the limited level of "intelligence" of the current systems.

In his seminal paper, March (1991) examined the organizational learning in humans and presented various features of, and relationships between, the essences of human organization learning: exploration of new possibilities and exploitation of old certainties. Allocating resources to these two capabilities represents the adaptiveness of the human organization. Based on this insight, a machine's intelligence can be considered as composed of the machine's capabilities of exploration, exploitation, and its ability to regulate the "resource" allocation between the two. This basic idea has been implemented in our research at two different layers. First, the reinforcement learning itself is based on the exploration-exploitation of the learned knowledge (i.e., a learner's current neural network) and the random choices. Second, the transfer learning allows the agent to exploit the previously learned experience (i.e., an expert's neural network obtained from the previous task context) and explore the new task context through learning and exploration. The long-term goal of this research is to develop an integrated transfer reinforcement learning technique that allows agents to learn from multiple task domains and exploit the learned knowledge in new task contexts for more effective learning and better task performance.

In this paper we focus on the robotic collision avoidance problem and investigate how transfer learning (Pan and Yang, 2010), in addition combined with deep reinforcement learning, can be applied to allow agents to exploit and explore in a much more complex task context. The rest of the paper is organized as follows. Section 2 provides a critical review of the relevant work in the areas of collision avoidance and machine learning and points out the gap in the literature. In Section 3, our proposed approach of transfer reinforcement learning is described in detail. Computational simulation based case studies are presented in Section 4 with the results being discussed in Section 5. Section 6 draws the conclusions and points to future research directions.

2 RELATED WORK

Collision avoidance problems have always attracted the attention of researchers in various fields: artificial intelligence, control theory, robotics, multi-agent system, etc. The traditional practice to achieve real-time obstacle avoidance was to create an artificial potential field (Khatib, 1986). Fahimi (2008) proposed harmonic potential functions and the panel method to address multi-robot obstacle avoidance problem in the presence of both static and dynamic obstacles. Mastellone et al. (2008) designed a controller for collision avoidance based on Lyapunov-type approach, and demonstrated the robustness of the system when the communication between robots was unreliable. Keller et al. (2016) designed a path planner for unmanned aircraft systems to provide surveillance by combining graph search and B-spline parametric curve construction, which could successfully navigate around obstacles and provide sufficient coverage. Tang and Kumar (2018) proposed the OMP+CHOP algorithm for a centralized multi-robot system, which was shown to be safe and complete, but at the cost of optimality.

In order for collision avoidance algorithms to be more adaptive and flexible in real world complex environment, learning capabilities of a multi-agent system have been developed. In recent years, deep learning has achieved tremendous success in various areas such as image recognition (Krizhevsky et al., 2012; Le et al., 2012), speech recognition (Hinton et al., 2012), automatic game playing (Mnih et al., 2013), self-driving (Bojarski et al., 2016) and so on. Deep learning algorithms can extract high-level features by utilizing deep neural networks, such as convolutional neural networks (CNNs) (Krizhevsky et al., 2012), multi-layer perceptrons and recurrent neural networks (RNNs) (LeCun, 2015). Scaling up deep learning algorithms is able to discover high-level features in a complex task. Dean et al. (2012) constructed a very large system which was able to train 1 billion parameters using 16000 CPU cores. Coates et al. (2013) scaled to networks with over 11 billion parameters using a cluster of GPU servers.

Minh et al. (2013) introduced deep learning algorithm using experience replay and CNNs to learn a Q function, which is able to play various Atari 2600 games better than human players. Experience replay allows an online learning agent to random sample batches from past experiences to update Q values, thus breaking the correlations between consecutive frames. By combining supervised learning and reinforcement learning approach, the group at Google DeepMind has further proven that their deep learning algorithm can outperform a world champion in the most challenging classic game Go (Silver et. al., 2016), which has extremely large number of possible configurations, and is difficult to evaluate board positions. Schaul et al. (2016) further developed a prioritized experience replay framework to sample more important transitions and learn more efficiently.

Chen (2016) developed a decentralized multi-agent collision avoidance algorithm based on deep reinforcement learning. Two agents were simulated to navigate toward their own goal positions and learn a value network which encodes the expected time to goal, and the solution was then generalized in multi-agent scenarios. Deep learning algorithms have been successful in achieving end-to-end learning. Dieleman and Schrauwen (2014) investigated whether it is possible to apply feature learning directly to raw audio signals by training convolutional neural networks. Traditionally content-based music information

retrieval tasks are resolved based on engineered features and shallow processing architectures, which relies on mid-level representations of music audio, e.g. spectrograms. The results showed that even though the end-to-end learning does not outperform the spectrogram-based approach, the system is able to learn automatically frequency decompositions and feature representations from raw audio.

Self-driving is a promising field which took off in the last few years and heavily relies on the advances in deep learning. Since self-driving cars always require a great deal of expensive and complex hardware, Yu et al. (2016) implemented a deep Q-learning algorithm using dataset (images) from real-time play of the game JavaScript Racer. In a recent published paper (Bojarski et al., 2016), a convolutional neural network is trained to map steering commands directly from raw pixels from camera input. The system automatically learned internal processing steps such as detecting useful road features with only the human steering angle as the training signal. This end-to-end learning approach is challenging in that it requires huge number of inputs and the advantage is that it releases the rely on the designer's prior domain knowledge.

Given a complicated task which is difficult to learn directly, transfer learning is a commonly used technique which can generalize previously learned experience and apply these experience into new tasks. Transfer learning refers to utilizing knowledge gained from source tasks to solve a target task. It is believed that in a reinforcement learning context, transfer learning can speed up the learning agent to learn a new but related task (i.e., target task) by learning source tasks first. Tayler and Stone (2007) introduced a transfer algorithm called Rule Transfer, which summarizes source task policy, modifies the decision list and generates a policy for the target task. Rule learning is well understood and human readable. The agent benefits from the decision list initially, and continues to refine its policy through target task training. It was shown that Rule Transfer could significantly improve learning in robot soccer using learned policy from a grid-world task.

Fernandez and Veloso (2006) proposed two algorithms to address the challenges of Policy Reuse in a reinforcement learning agent. The major components include an exploration strategy and a similarity function to estimate the similarity between past policies and new ones. Torrey (2006) introduced induction logic programming for analyzing previous experience of source task, and transferred rules for when to take actions. Through an advice-taking algorithm, the target task learner could benefit from outside imperfect guidance. A system AI2 (Advice via Induction and Instruction) for transfer learning in reinforcement learning was built, which creates relational transfer advice using inductive logic programming. Based on a human-provided mapping from source tasks to target tasks, the system was able to speed up reinforcement learning.

When utilizing deep neural networks in transfer learning, a base network on a base dataset is first trained on a source task, and the learned features are then transferred to the target network to be trained on a target dataset and task. If the size of target dataset is much smaller than source dataset, transfer learning is always beneficial to train a large target network. One common practice is to copy the first n layers of the base network to the first n layers of the target network while the remaining layers of the target network are randomly initialized and trained. Yosinski (2014) presented a way to determine whether a certain layer is general or specific. It was found that initializing a network with transferred features from almost any layers could boost the performance after fine-tuning to a new dataset. Dint et al. (2016) proposed a task-driven deep transfer learning framework for image classification, where the features and classifiers are obtained at the same time. Through pseudo labels for target domain, the system could transfer more discriminative information to the target domain. Parisotto et al. (2016) proposed a transfer reinforcement learning approach (Actor-Mimic) with two objectives – a policy regression objective and feature regression objective – to train a single policy network based on the insights of several experts, in order to mimic expert decisions for multi-task learning, which adopts the concept of policy distillation (Hinton et al, 2015). This approach has been tested to be successful in learning multiple Atari games.

Liu and Jin (2018) proposed a transfer reinforcement learning approach for autonomous collision avoidance where only static cases with high task-similarity were considered, and showed that transfer learning could boost learning as well as bring variance to the student learning process. To date there has been little literature aiming to combine deep reinforcement learning and transfer learning to solve robotic collision avoidance problems, because a) it is difficult to directly learn from raw pixel or distance sensory inputs, and b) it requires large amount of training data, which is not easy to generate in real life, c) the reward function is difficult to design. This research aims to close the gap between real world collision avoidance and deep learning by studying transfer reinforcement learning at a low inter-task similarity and developing transfer strategies accordingly.

3 TRL APPROACH

3.1 Deep reinforcement learning

Q-learning algorithm is a popular off-policy algorithm of reinforcement learning (RL), which assigns a value for each state-action pair and perform updates based on Bellman equation at iteration i (here s, a, r, s' denote the current state, action, immediate reward, and next state, respectively):

$$Q_{i+1}(s,a) = \mathbf{E}\left[r + \gamma \max_{a'} Q_i(s',a') \middle| s,a \right] \qquad (1)$$

We first start by summarizing the deep reinforcement learning (DRL) algorithm (Minh et al, 2013), which was originally applied to play Atari games. For an infinitely large state space such as a game window (a big array of pixel values), it is impossible to build a lookup Q-table and learn an optimal value for each state-action pair. DRL uses deep neural network as function approximator to approximate Q values. A Q-network

with weights θ_i can be trained by minimizing the loss function at each iteration *i*,

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

where $y_i = \mathbf{E}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1})\right]$ (3) is the target signal for iteration *i*. The gradient is calculated by the following:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbf{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right] \quad (4)$$

Various approaches have been proposed to stabilize and boost the learning process. Our neural network is built based upon the following three approaches:

- *Experience replay* (Minh et al, 2013)

The agents' experiences, $e_t = (s_t, a_t, r_t, s_{t+1})$, are stored into a *replay memory*, $D = e_1, e_2, ..., e_N$ (*N* is the capacity of the replay memory). Then mini-batches are randomly sampled from *D* and applied to Q-learning updates. The agent selects an action according to the ε-greedy policy. Experience replay increases data efficiency and breaks down the correlations between consecutive samples.

- *Double DQN* (van Hasselt et al., 2015)

The double DQN algorithm is able to solve the problem of overoptimistic value estimates, by separating the target network which is used for action evaluation, from the current network which is used for action selection. The agent's experience, is randomly assigned to update one of the two networks.

- *Dueling DQN* (Wang et al., 2016b)

In standard DQN, at each update of the Q values, only the value for one of the actions is updated whereas others remain untouched. Dueling DQN separates the Q value into a state value and action advantages, so that the state value is updated more frequently. Dueling network is useful to learn which states are or are not valuable, without having to explore each action for each state.

3.2 Transfer reinforcement learning

An expert network is obtained by training through a source task. The traditional can be expressed as: with probability ε of exploration and probability 1 - ε of exploitation, ε is often linearly decreasing through the learning process:

$$\varepsilon = \varepsilon_0\left(1 - \frac{t}{T_{\text{explr}}}\right) \quad (5)$$

where T_{explr} is the total exploration period during which ε is annealed to its minimum value.

In order to efficiently balance exploration and exploitation in the target task, a new transfer phase is added to the ε-greedy policy, which involves two crucial concepts: transfer belief and transfer period.

- Transfer belief β: how much confidence the agent puts in expert experience. More mathematically, transfer belief measures the probability of the agent picking transfer action suggested by the expert network.
- Transfer period Γ: how long the agent decision is influenced by expert network. After the transfer period, the agent will explore the environment based on its own using traditional ε-greedy policy. Transfer belief is linearly decreasing over the transfer period:

$$\beta = \begin{cases} \beta_0\left(1 - \frac{t}{\Gamma}\right), & \text{for } t \le \Gamma \\ 0, & \text{for } t > \Gamma \end{cases} \quad (6)$$

Then, the new ε_T - greedy policy is as following (see Figure 3.1):

(a) *Transfer*: With probability $p_1 = \beta$, pick the transfer action. The transfer action is defined as one of the **three** actions with the top **three** values of the expert network.
(b) *Exploration*: With probability $p_2 = \varepsilon(1-\beta)$, pick a random action.
(c) *Exploitation*: With probability $p_3 = (1-\varepsilon)(1-\beta)$, pick the best action calculated by the agent's current network.

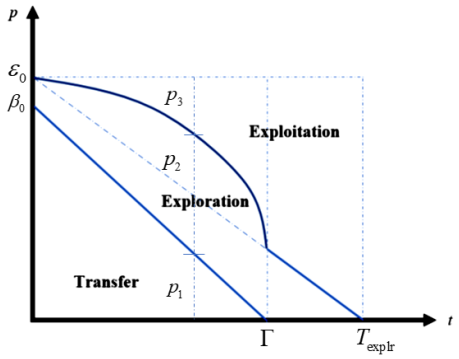

Figure 3.1: ε_T - greedy policy.

3.3 Agent learning behavior

A video game was created in Pygame to conduct case studies of transfer reinforcement learning on collision avoidance. The game environment consists a RL agent (green), obstacle and boundary walls (red), and goal area (orange), as shown in Figure 3.2.

- The state is defined as the pixel values of the game window. **Figure 3.2** shows an example.

- The action space is composed of seven actions with different combinations of linear and angular velocity, a_1 through a_7, as indicated in Table 1.
- The reward function is defined as:

$$r = \begin{cases} 200 & \text{if reach goal position} \\ -900 & \text{if hit any obstacle} \\ -1 & \text{otherwise} \end{cases}$$

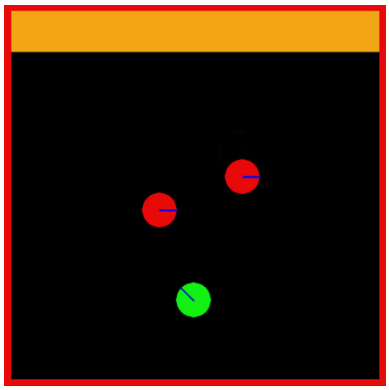

Figure 3.2: Game environment

Fig. 3.3 illustrates the proposed transfer reinforcement learning process. An **expert network** *Ne* is first obtained by training through the source task, which involves a single obstacle. In the target task, the learning agent (**the student**) follows ε_T - greedy policy to select actions with probabilities p_1, p_2, and p_3 as described in Subsection 3.2. After receiving a reward r_t from the environment, the agent stores the current experience e_t into the experience replay memory. The online network *Nc* is then updated by sampling mini-batches from the experience replay, as shown in **Fig. 3.3**.

Table 3.1 Agent actions

Action	v	ω
a_1	5	0.35
a_2	5	0.2
a_3	5	0.1
a_4	10	0
a_5	5	-0.1
a_6	5	-0.2
a_7	5	-0.35

3.4 Task complexity and similarity

In our previous work (Liu and Jin, 2018), we investigated transfer reinforcement learning between two similar tasks, i.e. from **single static** source task (Task A) to **multiple static** target task (Task B). There are several factors that contribute to the

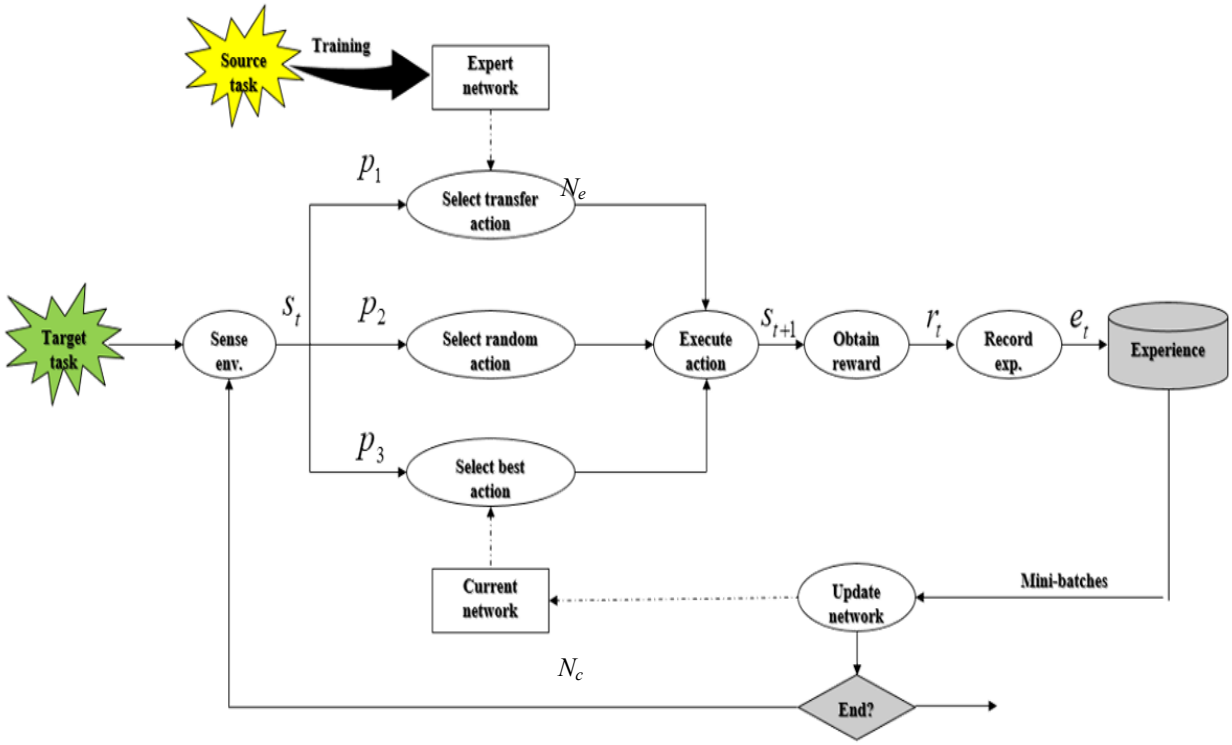

Figure 3.3: Agent learning behavior

overall complexity of each task (Table 3.2). For example, in the static case, the task complexity is determined by the shape and size of the obstacle, and possible locations of the obstacle. In the dynamic case where obstacles are moving, the obstacle speed may be constant, or changing over time bounded by obstacle's acceleration, which makes the environment much more complex. Tasks with similar complexity levels are believed to have higher similarity, whereas more difference on complexity levels makes the inter-task similarity lower. Our previous work on transferring from Task A to Task B is believed to have high inter-task similarity. In this research, we focus on transferring Task A to Task D, which has lower inter-task similarity comparatively. Even though Task D is the most complex among all task types, we keep the obstacle speed as constant for now to make it simpler and easier for agents to learn.

Table 3.2: Task complexity

	Static	**Dynamic (Moving)**
	• Shape/size • Location • …	• Speed • …
Single	**Task A** Low complexity	**Task C** Medium complexity
Multiple	**Task B** Medium complexity	**Task D** High complexity

4 CASE STUDIES

The objective of our case study is to test whether the optimal transfer belief / period from our previous work still works well in target tasks with lower similarity. If not, how to choose transfer belief and transfer period in the new task setting?

Two task situations are used for the case studies, namely "Source task – one static obstacle" and "Target task – two dynamic obstacles", as shown in Fig. 4.1. The obstacle is the same size as the agent. In the source task, at the beginning of each episode (game play), a random obstacle is generated within the dashed rectangle. In the target task, the obstacles are moving at a constant speed (20 pixels/time-step).

4.1 Choice of hyperparameters

The network structure is the same as the original DQN paper (Minh et al, 2013) with 84*84-pixel input and an output of 7 actions. The case studies were trained using Adam optimizer with a learning rate of 0.001. The discount factor γ is 0.99. In the source task the agent follows ε - greedy policy and in the target task the agent follows ε_T - greedy policy, with the exploration rate ε annealed from 1.0 to 0.1 over the first 1 million frames (1 frame = 1 state). The replay memory consists of 50, 000 most recent frames, and 50, 000 episodes were trained in total, (1 episode = from starting position to ending position). The initial transfer belief is chosen to be 0.9 or 0.5. The transfer period could be the first 300k or 700k frames. The choice of hyperparameters is summarized in **Table 4.1**.

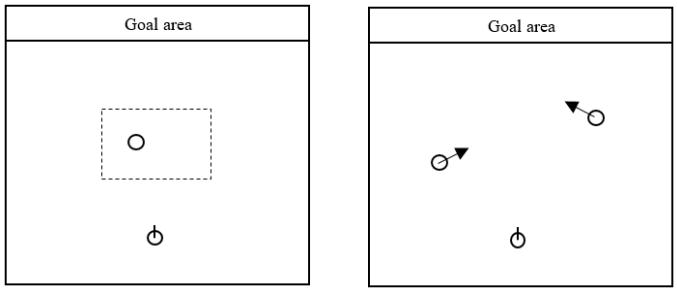

Figure 4.1: Source task (left): one static obstacle; Target task (right): two moving obstacles

Table 4.1 Case parameters

	Source task	**Target task**
Replay memory size	50,000	50,000
Mini-batch size	32	32
Discount factor	0.99	0.99
Learning rate α	0.001	0.001
Total training episodes	50,000	50,000
ε	1→ 0.1	1→ 0.1
Annealing frames	1 million	1 million
Transfer period (frames)	N/A	300k/700k
Initial transfer belief	N/A	0.5/0.9

5 RESULTS AND DISCUSSION

Each case result is obtained by running with 8 different random seeds (the shaded area in **Figure 5.1, 5.2, and 5.3.** The horizontal axis is the number of training episodes, and vertical axis represents the average reward). The darker line shows the average of these 8 runs. In each of these cases, the network is first initialized with the pre-trained weights from the source task. The baseline (green) is constructed by the agent exploring the environment using only ε -greedy policy.

Figure 5.1 shows the performance of $\beta_0 = 0.9$ and $\Gamma = 700k$ (orange) compared with the baseline. The choice of $\beta_0 = 0.9$ and $\Gamma = 700k$ comes from our previous study of transfer from Task A to Task B. As can be seen, the jumping start is still obvious due to the initial high transfer belief. But overall the boosting effect of transfer learning is rather small. The average performance almost overlaps with the baseline, which implies that the expert experience does not help that much in the new context where target task and source task have low similarity.

5.1 Decrease transfer period

The first option is to decrease transfer period Γ, from 700k to 300k, while keeping the initial transfer belief β_0 (0.9) the same (**Figure 5.2**). During the early stage, the performance is better than the baseline. However, after the transfer period, the learning variance starts to grow. Though the maximum performance is still better than baseline, many students perform worse than the baseline. The average performance is slightly higher than baseline, but should not be considered as improvement.

5.2 Decrease transfer belief

The second option is to decrease initial transfer belief β_0, from 0.9 to 0.5, which means in the beginning the agent has 50% chance of picking transfer action suggested by the expert network (this probability is linearly decreasing to 0 until the end of transfer period). As is shown in **Figure 5.3**, the jump-start effect is less obvious compared to β_0 = 0.9. Additionally, the starting variance is higher than β_0 = 0.9. However, after the transfer period, many students perform much better than the baseline. The average performance (red) converges to the optimal much earlier than baseline (blue).

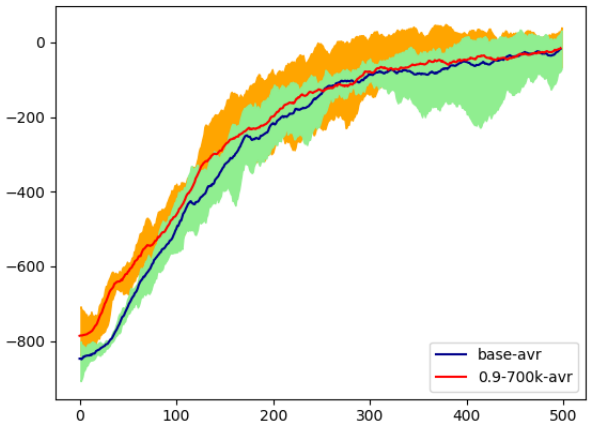

Figure 5.1: Performance of β_0 = 0.9 and Γ = 700k

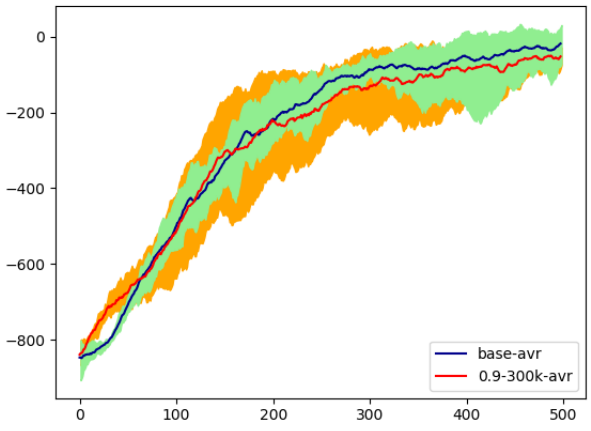

Figure 5.2: Performance of β_0 = 0.9 and Γ = 300k

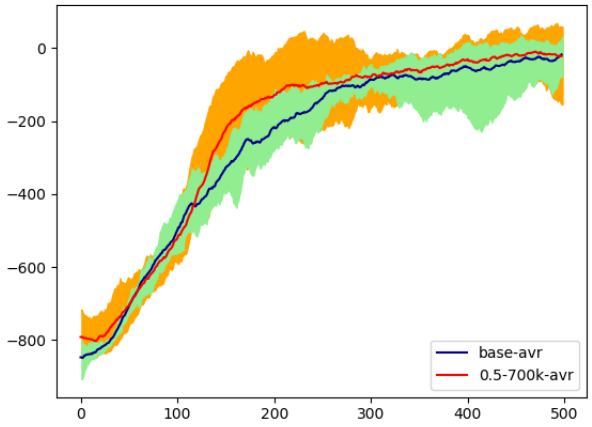

Figure 5.3: Performance of β_0 = 0.5 and Γ = 700k

6 CONCLUSIONS AND FUTURE WORK

In this research, both transfer learning and deep reinforcement learning have been adopted to solve the problem of collision avoidance. The proposed TRL approach is tested in a context where source task and target task share low similarity. Based on the case studies, our findings are summarized as following:

- As a designer, we need to carefully choose transfer belief and transfer period. Most of the time transfer learning will boost learning in a new target task. However, some bad choices of transfer belief and transfer period can bring negative transfer.
- One set of transfer belief and transfer period which works well in a certain target task might not work as well in another target task that has different similarity. In fact, sticking with a fixed transfer belief and transfer period is more likely to cause negative transfer – either slows down the learning process or increases the learning variance.
- If two tasks have low similarity, it is better to decrease initial transfer belief and keep a relatively longer transfer period, due to two reasons: first, expert experience from the source task does not help the agent as much in the target task; second, transfer period being too short makes the learning vary a lot among different student networks and many students perform even worse than the baseline without transfer.
- Compared with lower initial transfer belief (β_0 = 0.5), higher initial transfer belief (β_0 = 0.9) has more jump-start effect and makes the variance rather small during the early stage.

Our ongoing research is to quantitatively measure task complexity levels and systematically determine the inter-task similarity accordingly. We also plan to design an automated mechanism to choose (initial) transfer belief and transfer period given different inter-task similarity.

7 ACKNOWLEDGMENTS

This paper was based on the work supported by the Monohakobi Technology Institute (MTI) and Nippon Yusen Kaisha (NYK). The authors are grateful to the MTI team for their discussions and insights on this research.

8 REFERENCES

1. Alonso-Mora, Javier, et al. "Optimal reciprocal collision avoidance for multiple non-holonomic robots." *Distributed Autonomous Robotic Systems*. Springer, Berlin, Heidelberg, 2013. 203-216.
2. Arnold, Andrew, Ramesh Nallapati, and William W. Cohen. "A comparative study of methods for transductive transfer learning." *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference*. IEEE, 2007.
3. Bahadori, Mohammad Taha, Yan Liu, and Dan Zhang. "A general framework for scalable transductive transfer learning." *Knowledge and information systems* 38.1 (2014): 61-83.
4. Bojarski, Mariusz, et al. "End to end learning for self-driving cars." *arXiv preprint arXiv:1604.07316* (2016).
5. Brunn, Paul. "Robot collision avoidance." *Industrial Robot: An International Journal* 23.1 (1996): 27-33.
6. Casanova, D., C. Tardioli, and A. Lemaître. "Space debris collision avoidance using a three-filter sequence." *Monthly Notices of the Royal Astronomical Society* 442.4 (2014): 3235-3242.
7. Chen, Jim X. "The evolution of computing: AlphaGo." *Computing in Science & Engineering* 18.4 (2016): 4-7.
8. Churchland, Patricia S., and Terrence J. Sejnowski. *The computational brain*. MIT press, 2016.
9. Coates, Adam, et al. "Deep learning with COTS HPC systems." *International Conference on Machine Learning*. 2013.
10. Chen, Yu Fan, et al. "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning." *Robotics and Automation (ICRA), 2017 IEEE International Conference*. IEEE, 2017.
11. Dean, Jeffrey, et al. "Large scale distributed deep networks." *Advances in neural information processing systems*. 2012.
12. Dieleman, Sander, and Benjamin Schrauwen. "End-to-end learning for music audio." *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014.
13. Ding, Zhengming, Nasser M. Nasrabadi, and Yun Fu. "Task-driven deep transfer learning for image classification." *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016.
14. Eleftheria, Eliopoulou, Papanikolaou Apostolos, and Voulgarellis Markos. "Statistical analysis of ship accidents and review of safety level." *Safety science* 85 (2016): 282-292.
15. Fahimi, Farbod, Chandrasekhar Nataraj, and Hashem Ashrafiuon. "Real-time obstacle avoidance for multiple mobile robots." *Robotica* 27.2 (2009): 189-198.
16. Fernández, Fernando, and Manuela Veloso. "Probabilistic policy reuse in a reinforcement learning agent." *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, 2006.
17. Frommberger, Lutz. "Learning to behave in space: A qualitative spatial representation for robot navigation with reinforcement learning." *International Journal on Artificial Intelligence Tools* 17.03 (2008): 465-482.
18. Fujii, Teruo, et al. "Multilayered reinforcement learning for complicated collision avoidance problems." *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*. Vol. 3. IEEE, 1998.
19. Goerlandt, Floris, and Pentti Kujala. "On the reliability and validity of ship–ship collision risk analysis in light of different perspectives on risk." *Safety science* 62 (2014): 348-365.
20. Hameed, Saad, and Osman Hasan. "Towards autonomous collision avoidance in surgical robots using image segmentation and genetic algorithms." *Region 10 Symposium (TENSYMP), 2016 IEEE*. IEEE, 2016.
21. Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).
22. Jin, Y. and T. Koyama, "On the Design of Marine Traffic Control System (1st Report)", in *Journal of the Society of Naval Architects of Japan*, Vol. 162, pp.183-192, December 1987
23. Keller, James, et al. "Obstacle avoidance and path intersection validation for UAS: A B-spline approach." *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*. IEEE, 2016.
24. Khatib, Oussama. "Real-time obstacle avoidance for manipulators and mobile robots." *Autonomous robot vehicles*. Springer, New York, NY, 1986. 396-404.
25. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.
26. Kuderer, Markus, Shilpa Gulati, and Wolfram Burgard. "Learning driving styles for autonomous vehicles from demonstration." *Robotics and Automation (ICRA), 2015 IEEE International Conference*. IEEE, 2015.
27. Le, Quoc V. "Building high-level features using large scale unsupervised learning." *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013.
28. LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." *Neural computation* 1.4 (1989): 541-551.
29. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *Nature* 521.7553 (2015): 436.
30. Liu, X. and Jin, Y. Design of Transfer Reinforcement Learning Mechanisms for Autonomous Collision

Avoidance. *8th International Conference on Design Computing and Cognition*, 2018.
31. Machado, Toni, et al. "Multi-constrained joint transportation tasks by teams of autonomous mobile robots using a dynamical systems approach." *Robotics and Automation (ICRA), 2016 IEEE International Conference*. IEEE, 2016.
32. March, James G. "Exploration and exploitation in organizational learning." *Organization science* 2.1 (1991): 71-87.
33. Mastellone, Silvia, et al. "Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments." *The International Journal of Robotics Research*27.1 (2008): 107-126.
34. Matarić, Maja J. "Reinforcement learning in the multi-robot domain." *Robot colonies*. Springer, Boston, MA, 1997. 73-83.
35. Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv: 1312.5602* (2013).
36. Mukhtar, Amir, Likun Xia, and Tong Boon Tang. "Vehicle detection techniques for collision avoidance systems: A review." *IEEE Transactions on Intelligent Transportation Systems* 16.5 (2015): 2318-2338.
37. Ohn-Bar, Eshed, and Mohan Manubhai Trivedi. "Looking at humans in the age of self-driving and highly automated vehicles." *IEEE Transactions on Intelligent Vehicles* 1.1 (2016): 90-104.
38. Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." *IEEE Transactions on knowledge and data engineering* 22.10 (2010): 1345-1359.
39. Schaul, Tom, et al. "Prioritized experience replay." *arXiv preprint arXiv:1511.05952* (2015).
40. Shiomi, Masahiro, et al. "Towards a socially acceptable collision avoidance for a mobile robot navigating among pedestrians using a pedestrian model." *International Journal of Social Robotics* 6.3 (2014): 443-455.
41. Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529.7587 (2016): 484-489.
42. Tang, Sarah, and Vijay Kumar. "A complete algorithm for generating safe trajectories for multi-robot teams." *Robotics Research*. Springer, Cham, 2018. 599-616.
43. Taylor, Matthew E., and Peter Stone. "Cross-domain transfer for reinforcement learning." *Proceedings of the 24th international conference on Machine learning*. ACM, 2007.
44. Torrey, Lisa, et al. "Skill acquisition via transfer learning and advice taking." *European Conference on Machine Learning*. Springer, Berlin, Heidelberg, 2006.
45. Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." *AAAI*. Vol. 16. 2016.
46. Wang, Fei-Yue, et al. "Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond." *IEEE/CAA Journal of Automatica Sinica* 3.2 (2016): 113-120.
47. Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." *arXiv preprint arXiv:1511.06581*(2015).
48. Watkins, C. Learning from delayed rewards, 1989. Doctoral dissertation, University of Cambridge.
49. Yang, I. B., S. G. Na, and H. Heo. "Intelligent algorithm based on support vector data description for automotive collision avoidance system." *International journal of automotive technology* 18.1 (2017): 69-77.
50. Yosinski, Jason, et al. "How transferable are features in deep neural networks?." *Advances in neural information processing systems*. 2014.
51. Zou, Xueyi, Rob Alexander, and John McDermid. "On the validation of a uav collision avoidance system developed by model-based optimization: Challenges and a tentative partial solution." *Dependable Systems and Networks Workshop, 2016 46th Annual IEEE/IFIP International Conference on*. IEEE, 2016