Design of Transfer Reinforcement Learning Mechanisms for Autonomous Collision Avoidance

Xiongqing Liu and Yan Jin
University of Southern California, USA

Abstract
It is often hard for a reinforcement learning (RL) agent to utilize previous experience to solve new similar but more complex tasks. In this research, we combine the transfer learning with reinforcement learning and investigate how the hyper-parameters of both transfer learning and reinforcement learning impact the learning effectiveness and task performance in the context of autonomous robotic collision avoidance. A deep reinforcement learning algorithm was first implemented for a robot to learn, from its experience, how to avoid randomly generated single obstacles. After that the effect of transfer of previously learned experience was studied by introducing two important concepts, *transfer belief*—i.e., how much a robot should believe in its previous experience—and *transfer period*—i.e., how long the previous experience should be applied in the new context. The proposed approach has been tested for collision avoidance problems by altering transfer period. It is shown that transfer learnings on average had ~50% speed increase at ~30% competence levels, and there exists an optimal *transfer period* where the *variance* is the lowest and learning *speed* is the fastest.

1 Introduction

Collision avoidance is a common research topic in many industrial fields. In the area of robotics, research has been focused on issues related to how vehicle robots avoid obstacles as well as each other (Shiomi et al, 2014) and how assembly robots avoid interferences among its own arms or with those of others (Hourtash et al, 2016). In transportation. Self-driving cars must be able to avoid obstacles and other vehicles in various situations (Mukhtar et

Design Computing and Cognition DCC'18. J.S. Gero (ed), pp. 339-358. © Springer 2018

al, 2015). In the shipping industry, collision avoidance can be highly difficult, when water areas are congested, due to the large ship inertia causing immovability (Goerlandt & Kujala, 2014). 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.

Approaches to solving collision avoidance problems can be divided into two large categories, one is vehicle control system (Machado et al, 2016), relying on traditional control theories and intelligent systems approach, and the other traffic system development. 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 through supervised deep learning and from machines' own experiences through reinforcement learning. The reinforcement learning approach allows an agent to learn from its 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., Alpha-Go (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 machine learning systems, including AlphaGo, is that they can only function 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., the agent'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 to deep

reinforcement learning, can be applied to allow agents to exploit and explore across task contexts. In the following, Section 2 provides a critical review of the relevant work in collision avoidance and machine learning and points out the gap in the literature. Section 3 describes our proposed approach of transfer reinforcement learning in detail. Computer 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 (2015) 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.

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), 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.

Mnih et al. (2013) introduced deep learning algorithm using experience replay and CNNs to learn a Q function, tha can play various Atari 2600 games better than human players. Experience replay allows an learning agent to randomly sample batches from past experiences to update Q values, thus breaking the correlations between consecutive frames. By combining supervised learning and reinforcement learning, a group at 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). 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) applied feature learning directly to raw audio signals by training convolutional neural networks. The results showed that the system learns automatically frequency decompositions and feature representations from raw audio.

Self-driving is a promising field that 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. This end-to-end learning approach is challenging in that it requires huge number of inputs and the advantage is that it releases the reliance on the designer's prior domain knowledge.

Transfer learning refers to utilizing knowledge gained from source tasks to solve a target task. In a reinforcement learning context, transfer learning can potentially 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* can 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. The PRQ-learning

algorithm probabilistically bias an exploration learning process by using a *policy library*. In the second algorithm called PLPR, the policy library is created when learning new policies and reusing past policies.

Torrey (2006) introduced the 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 AI^2 (Advice via Induction and Instruction) for transfer learning in reinforcement learning was built, which creates relational *transfer advice* using inductive logic programming.

In transfer learning within deep neural networks, a base network on a base dataset and task is first trained, and the learned features are then transferred to the target network to be trained on a target dataset and task, commonly by copying the first *n* layers of the base network to the first *n* layers of the target network. A task-driven deep transfer learning framework for image classification was designed (Ding et al., 2016), where the features and classifiers are obtained at the same time. Parisotto et al. (2016) proposed a transfer reinforcement learning approach (Actor-Mimic) to mimic expert decisions for multi-task learning, which adopts the concept of *policy distillation* (Hinton et al, 2015).

To date there has been little literature aiming to combine deep reinforcement learning and transfer learning to solve robotic collision avoidance problems, because (1) it is difficult to directly learn from raw pixel or distance sensory inputs, and (2) it requires large amount of training data, which is not easy to generate in real life. This research aims to close the gap between real world collision avoidance and deep learning by proposing a combined transfer & reinforcement learning approach to learn a new task more efficiently.

3 A Transfer & Reinforcement Learning Approach

Before moving into details of the mechanism for collision avoidance, we first introduce the basic idea and our overall goal of research on integrated machine learning for developing intelligent systems.

Reinforcement learning has the advantage of learning from the agent's own experience and the agent learns to choose actions at any state to maximize the total rewards by interacting with the environment. Although reinforcement learning allows agents to acquire collision avoidance skills (Mataric, 1998; Fujii et al, 1998; Frommberger, 2008), one challenge is that it requires a large amount of training data, which is usually hard to obtain in

real life considering the cost of building physical systems and conducting experiments.

On the other hand, recent progress in self-driving car research (Bojarski, 2016; Ohn-Bar & Trivedi, 2016) and deep learning, e.g., Alpha-Go (Chen, 2016; Churchland & Sejnowski, 2016; Wang et al, 2016), have demonstrated that the experience of human "experts" represents a highly valuable source of intelligence and can be learned by machines through deep learning. However, in many situations, the access to human expertise data can be very limited, since it is difficult, if not impossible, to acquire human experience data in all possible situations. How to effectively and efficiently combine human expertise with machine self-learning remains to be a challenge.

In this research, we consider that a machine's "intelligence" is dependent on three fundamental capabilities:

- First, it must be able to "exploit" the existing knowledge or expertise to the maximum extent so that all the known situations can be dealt with. This capability corresponds to transfer learning at a macro scale and deep learning mechanisms at a micro scale.
- Second, the machine must be able to "explore" the unknown territories and develop new knowledge or expertise from its own experience. Reinforcement learning is a candidate for this capability.
- Lastly, depending on the level of dynamics of the task domain and environment, the machine must be able to "adapt" the ratio of exploitation over exploration in order to stay effective. More dynamic or changeable domains require more exploration. Human design or meta-level learning mechanisms are needed to deal with this issue.

Our long-term goal is to develop an integrated machine learning technology that can (1) learn from multiple experts from diverse domains, (2) apply the learned expertise to explore new domains (e.g., requiring multiple domain expertise, or more complex), and (3) manage its own learning processes (i.e., exploitation and exploration) according to the change in task domains. The "domains" can be knowledge domains, such as mechanical design, and technical domains, such as robotic (e.g., robot, car, ship) collision avoidance. Our current focus is on technical domains.

As the first step in the research, we seek to develop an integrated learning mechanism that can take advantage of existing steering experience from either humans or other robots to learn about actions in new and more complex situations. More specifically, we propose a *transfer reinforcement learning*, or TRL for short, approach built on deep reinforcement learning algorithms. By combining the experience from the "expert", the agent can reduce trial time and learn about more complex tasks faster.

3.1 Deep Reinforcement Learning

The attempt was made to use reinforcement learning algorithms to train the system so that it automatically learns to solve tasks only from sensory inputs and a scalar reward signal. However, it was difficult to collect the sufficient amount of data as the training input, especially in real life, by only relying on sensory inputs. In addition, the state-action space is always continuous which makes it impractical to build a look-up Q-table. To overcome the curse of dimensionality, deep neural networks are used as functional approximators to replace the Q-table and approximate Q values.

We began this research by implementing the deep reinforcement learning algorithm with experience replay as proposed in (Mnih, 2013). We first consider standard Q-learning (Watkins, 1989) which can be formulated as a tuple of $\langle S, A, P, R, \gamma \rangle$. S is the state space, which consists of all the agent's possible states in the environment. A is the action space consisting of all the possible actions that the agent can take. P is the transition matrix (usually unknown in a model-free environment), R is the reward function, and γ is the discount factor. At any given time t, the agent's goal is to maximize its future discounted return $R_t = \sum_{t'}^{T} \gamma^{t'-t} r_{t'}$, where T is the time when the game terminates. Like many other reinforcement learning algorithms, the agent estimates at each time step the action-value function $Q(s,a)$, using Bellman equation as an update. Such value iteration algorithms converge to the optimal value function.

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

In order to adapt to tasks involving infinitely large state/action space where building the Q table is impractical, deep Q learning with experience replay uses a neural network as a function approximator (Q-network). A Q-network with weights θ_i can be trained by minimizing the loss function $L_i(\theta_i)$ at each iteration i,

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

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

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

The deep Q learning algorithm utilizes a technique called *experience replay* where 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.

Various approaches have been addressed to stabilize learning process, such as Deep Q network (DQN) (Mnih et al, 2013), double DQN (van Hasselt et al., 2015) and dueling DQN (Wang et al., 2016b). In this research, our base network is built by combining these three approaches.

3.2 Transfer Reinforcement Learning

The goal of transfer learning is to transfer "expert" knowledge into a learning agent (student) for new tasks which are more complex. The expert network is first obtained by training through the source task, and then used for initialization in the student's network for the target task. In order to utilize the expert experience more efficiently, a new *transfer phase* is added to the traditional ε-greedy policy (**Fig. 1**), where the agent selects *transfer action* according to the expert network. The transfer action is defined as one of the **three** actions with the top **three** values of the expert network. This new policy is called ε_T-greedy policy, which is defined as the following:

(a) *Transfer*: With probability $p_1 = \beta_0\left(1 - \frac{t}{T_{tran}}\right)$, the agent selects the transfer action—i.e., the action suggested by the expert neural network. T_{tran} shown in **Fig.1** is the *transfer period*, during which the agent is influenced by the expert network (transfer period is shorter than the exploration period T_{expl}, where ε is annealed close to 0.1); β_0 is the initial *transfer belief*, which measures how much confidence the agent puts in the expert knowledge.

(b) *Exploration*: With probability $p_2 = \varepsilon(1 - p_1)$, the agent selects a random action.

(c) *Exploitation*: With probability $p_3 = (1 - \varepsilon)(1 - p_1)$, the agent selects the current best action produced by its own learned knowledge/ network.

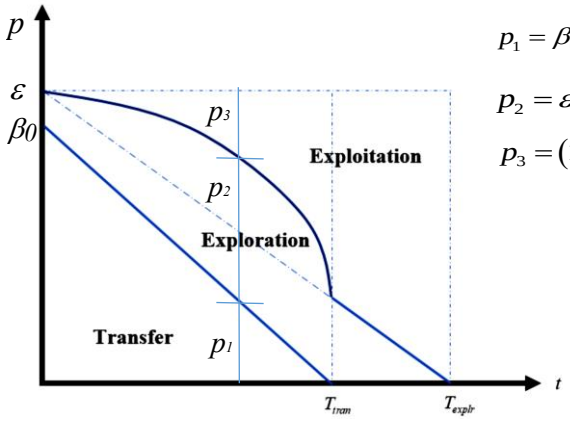

Fig. 1 Exploration/exploitation with transfer

3.3 Agent Learning Behavior

A computer game environment was created to conduct case studies for transfer reinforcement learning for collision avoidance. The game environment consists of a learning agent (green), static obstacle (red), and a goal area (orange), as shown in **Fig. 2**. The simpler *source task* has only a single static obstacle whereas the more complex *target task* always has two obstacles. The obstacle is randomly generated at the beginning of each collision avoidance episode.

- The state in the case studies is defined as the pixel values of the game window. **Fig. 2** shows an example.
- The action space is composed of seven actions, a_1 through a_7, as indicated in **Table 1**.
- The reward function is defined as:

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

Fig. 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 agent follows ε_T - greedy policy to select actions with probabilities p_1, p_2, and p_3 as described in Subsection 3.2.

Table 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

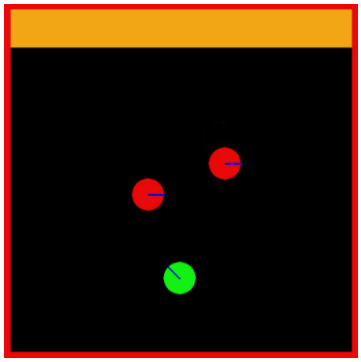

Fig. 2 Environment setup

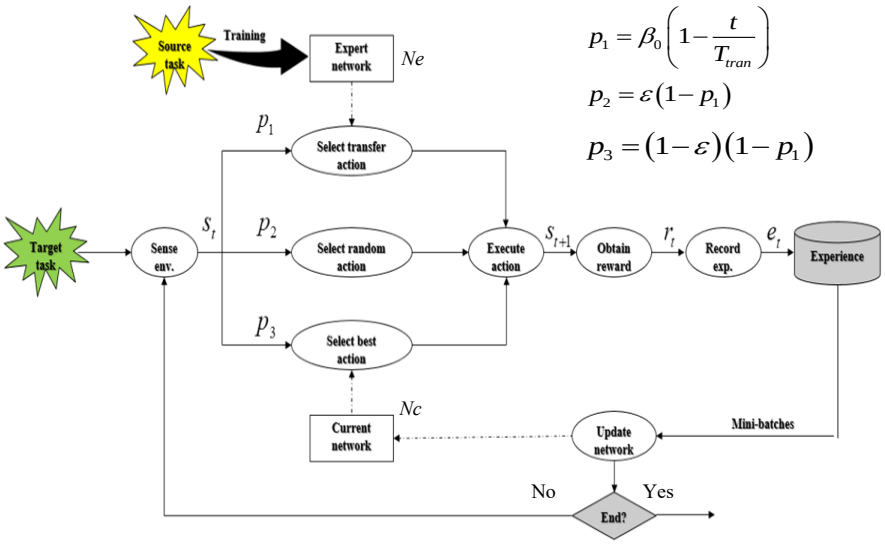

Fig. 3 Agent learning behavior

After receiving a reward r_t from the environment, the agent stores the current experience e_t into the experience replay memory. The currently learned network N_c is then updated by sampling mini-batches from the experience replay, as shown in **Fig. 3**.

4 Case studies

4.1 Collision Avoidance Game System Architecture

The collision avoidance game system consists of two modules: a visualization module (Pygame) and a machine learning module (TensorFlow). The visualization module creates graphical display for the system, where it reads the current environment state and simulates kinematics and dynamics. After taking some action, the agent will receive a reward, based on which a replay memory is constructed and sent to the machine learning module. TensorFlow deals with the heavy lifting to sample batches of experience and update the network weights, and then sends the updated weights back to the visualization module, as shown in **Fig. 4**.

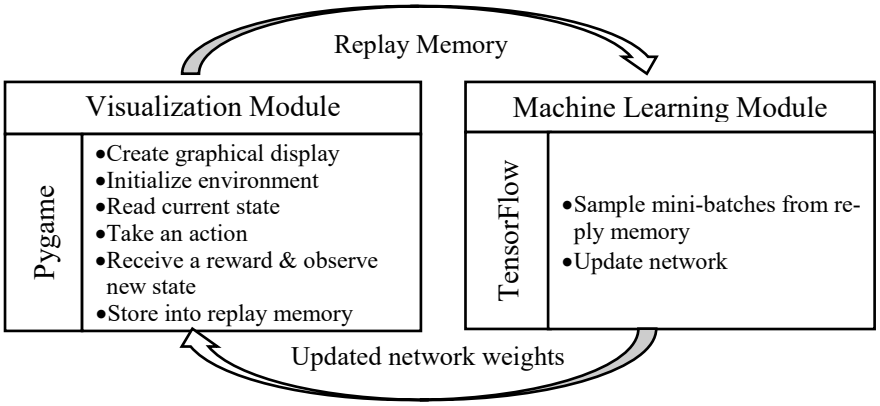

Fig. 4 Collision avoidance game system architecture

4.1 Case Parameters

Two task situations are used for the case studies, namely "Source task – one obstacle" and "Target task – two obstacles", as shown in **Fig. 5**. As indicated in the figure, the source task has a smaller obstacle area and only one obstacle can randomly appear in the obstacle area. The target task situation, however, has a much wider obstacle area and there are always two obstacles that can appear in any random relative positions within the large obstacle area.

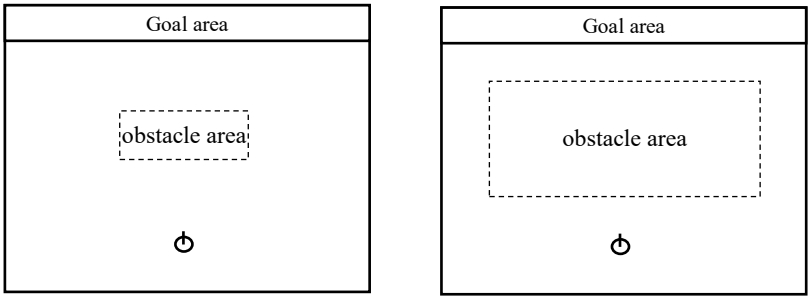

Fig. 5. Case study task situations

The network structure is the same as the original DQN paper (Mnih et al, 2013) with an array of 84*84 pixels input and an output of 7 actions. All our case studies were trained using Adam optimizer with a learning rate of 0.001. The discount factor γ is 0.99. The agent follows either ε-greedy policy in the source task or ε_T-greedy policy in the target task with ε annealed from 1.0 to 0.1 over the first 1 million frames, with 1 frame = 1 state. The replay memory consists of 50, 000 recent frames, and 50, 000 episodes were trained in total, with 1 episode = 1 game play. The transfer period could be the first 150k, 300k, 700k, or 1 million frames. The choice of hyper-parameters is summarized in **Table 1**.

Table 1 Hyper-parameters

	Source task One-obstacle	**Target task Two-obstacle**
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	150k/300k/700k/1m
Initial transfer belief	N/A	0.9

5 Results and Discussion

After an expert network is obtained by training through the source task with single obstacle, the agent is then given a more complex target task, which has two obstacles and a larger obstacle field. The results of learning efficiency and effectiveness are shown in **Figs. 6, 7 & 8**. All the curves in **Fig. 6** are the average of 10 learning performances by running 10 random seeds. The curves are also smoothed using exponential smoothing with the dumping factor set to 0.9. In all three figures, the unit of the x-axis is the number of 100 episodes. Each episode is defined as the period from agent starting movement to arriving at the goal, as shown in **Fig. 5**.

The colors in the figures indicate different lengths of the transfer period, which is measured in number of frames. For example, the blue line in **Fig. 6** shows the performance of transfer learning with transfer period = 300K frames. Roughly, 1 million frames = 115 (x100) episodes. The two red colored baseline cases, discussed below, do not use ε_T -greedy decision policy.

The y-axis of **Figs. 6 & 7** is the total reward value. Since the reward function is set to heavily penalize the collision with the obstacle and very small positive values are for reaching the goal, the final value of the total reward is close to zero. In Figure 8, the y-axis shows the standard deviation of multiple learning runs at different number of episodes, measured as total reward value.

5.1 Baseline Cases

For the purpose of comparison, we established two baseline cases. The first baseline case is for an agent to learn about the "target task – two obstacles" by "*bootstrap*"—i.e., the neural network is randomly initialized. The dark red lines shown in **Figs. 6 & 7** indicate the learning performance of this baseline case. As the figures show, starting from scratch requires more time for the agent to learn about the task. Especially, it takes much longer training for the agent to become capable of dealing with the two-obstacle collision avoidance.

Another baseline case is "*copy expert*"—i.e., the weights of the expert network learned from the source task are copied into the learning agent as the initial neural network for the "target task." After initialization, the agent starts its regular reinforcement learning: the copied expert network weights are updated by following the ε -greedy policy (i.e., ε starts from 1 and annealed to 0.1) to select actions. The red line shown in **Figs. 6 & 7** indicates the learning performance of this baseline case.

5.2 Learning Speed

Baseline case *bootstrap*: As shown by the dark-red line in **Fig. 6**, without any input from the expert knowledge, it takes much longer for the agents to learn about the *target task*. Huge lag appears until around 11K frames point. However, it catches up very fast after that point. The final learning effectiveness within the 50K frames range is inferior.

Baseline case *copy expert*: In this case, as shown of the red color in **Fig.6**, the starting point for the learning agent is a complete copy of the expert network. Since immediately after the learning process starts, the "expert network" will be updated by following ε-greedy policy, the "expert supervision" does not really exist. As a result of *copy-expert*, the learning picks up faster than *bootstrap* case with almost the constant speed. We believe that the difference in learning speed between these two baseline cases indicates the level of similarity of the source task and target task domain. A detailed discussion of the similarities between source and target task domains will be presented in a separate publication (Liu and Jin, 2018).

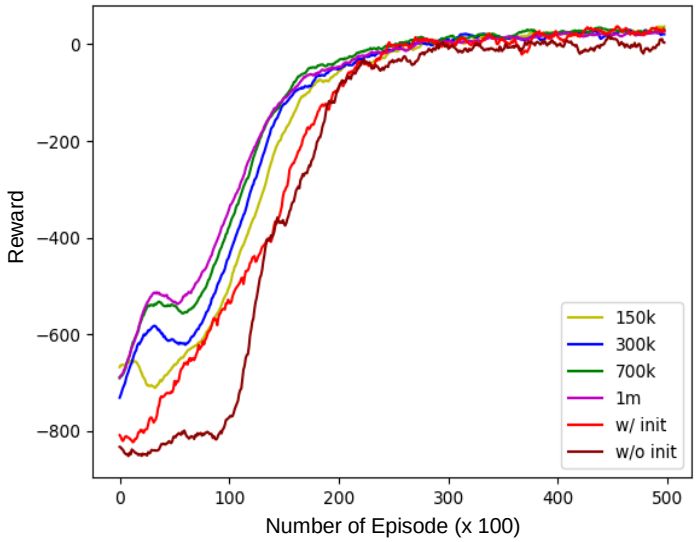

Fig. 6 Average learning performance of each transfer period

Transfer reinforcement learning (TRL) cases with *varying transfer period*: Our primary simulation runs of TRL processes have revealed that the transfer period plays a key role in affecting learning speed. **Fig. 6** illustrates the learning performance of varying transfer period from 150K, 300K, 700K, to 1M frames with yellow, blue, green and pink colors, respectively. As shown in **Fig. 1**, shorter transfer period T_{trans} means shorter period of

expert supervision—i.e., to use expert network *Ne* to select actions (also see **Fig. 3**). From a learning speed point of view, the results in **Fig. 6** indicate that longer transfer periods lead to better learning performance, with the effect diminishing as it becomes sufficiently long (after 700K frames). When the transfer period is getting closer to 1 million frames—i.e., the annealing time when ε decreases to 0.1—the performance decreases. Comparing with the two baseline cases discussed above, the positive impact of *expert supervision* is considerably large, especially until the 200K episodes range.

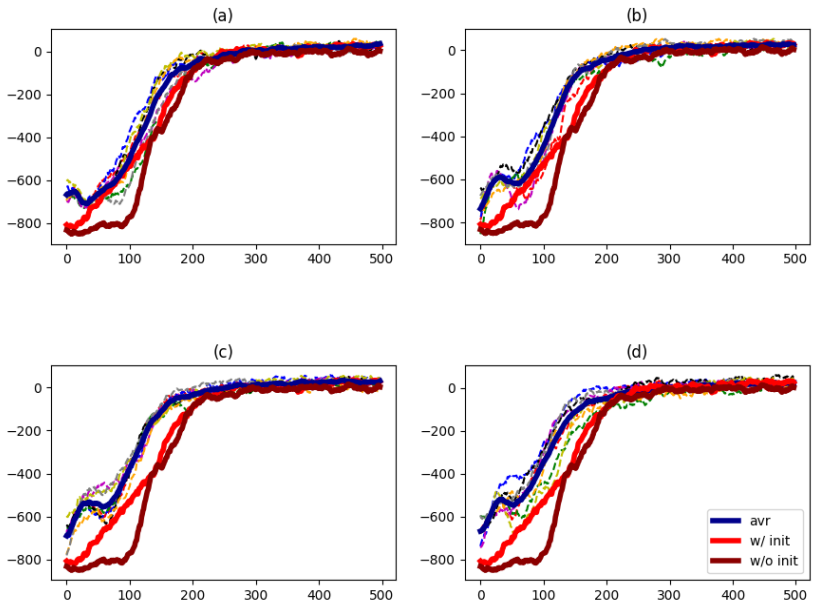

Fig. 7 Different transfer period (a) 150K frames, (b) 300K frames, (c) 700K frames, and (d) 1 million frames

5.3 Learning Variance

In addition to learning speed, we identified the variance as an important measure of learning performance since in most intelligent engineering systems, the consistency of learning performance is very much demanded. **Fig. 7** illustrates the learning variance multiple learning runs with different transfer periods of first 150K, 300L, 700K, 1M frames. Each color represents an independent trial. Each transfer period has 10 trials in total. The red curves are the two baseline cases. The standard deviation of each transfer period case before convergence (t from 0 to 200) is shown in **Fig. 8**. It can be seen

that the variances of different transfer periods share a similar pattern: decreases at beginning, then increases, and finally decreases again as the learning converges. The width of the exploration (see **Fig. 1**) played a role in determining such a pattern.

A careful examination of Fig. 8 indicates that the overall variances are larger for both short transfer period case (150K frames) and long transfer period case (1M frames), while the 300K-700K transfer period cases appear to have less variance for different learning trials, exhibiting more consistent learning performance of the system. Further research is needed to investigate this interesting phenomenon.

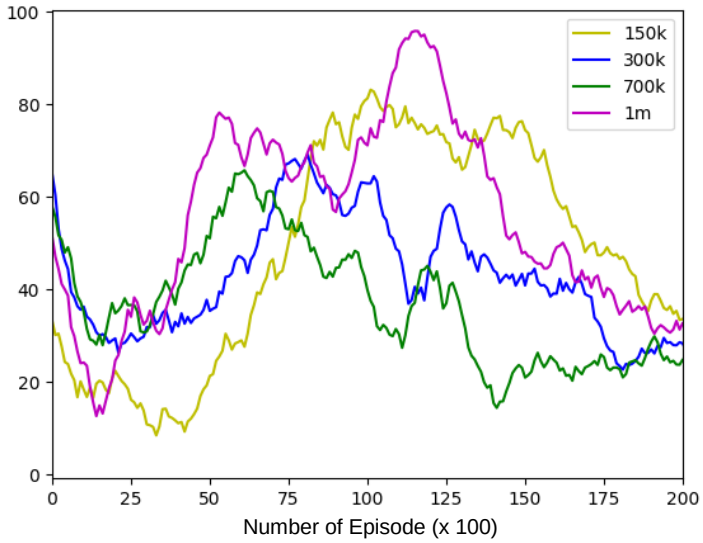

Fig. 8 Standard deviation plot of various transfer periods

6 Conclusions and Future Work

Collision avoidance is a common research topic in various industrial fields. Recent progress in machine learning has made it possible to train robots or agents to acquire collision avoidance knowledge. Although in the engineering research community, design is still focused on *static* and *dynamic, mechanical* and *structural* systems, future demands on intelligent engineering systems call for methods for designing *intelligent and learning* systems. In this research, we approach the problem of collision avoidance from an in-

telligent and learning systems perspective. By considering machine intelligence as the capabilities of exploitation and exploration together with adaptation, we developed a transfer reinforcement learning approach that can be tuned to exploit past experience of human experts and other robots and explore the new domain through deep reinforcement learning. Following is a summary of our findings.

- The proposed transfer reinforcement learning approach has been tested in a game environment and proved useful to solve similar complex collision avoidance tasks.
- The transfer period is a crucial component that needs to be adjusted. Our transfer learning scheme has two effects: learning *speed* and *variance*. Compared to the *bootstrap* case, the *copy expert* strategy performed better. Comparing with *bootstrap,* the transfer learnings on average had a ~50% increase at ~25% competence level and ~30% increase at 75% competence level. As transfer period increases, the learning speed increases. However, transfer period being too long may slow down the learning, but still faster than the baselines.
- The standard deviation plot shows that variance starts to decrease, and then increases, and finally decreases as learning converges. The longer transfer period, the earlier variance starts to increase. As learning proceeds, either short or long transfer period leads to high variance, whereas medium transfer period has low variance.
- There exists an optimal length of transfer period (700K frames) when the variance is low, and learning is fast. This optimal transfer period is believed to be task-dependent, which is relevant to the inter-task similarity of source and target task.

Our ongoing research investigates task similarities and transfer strategies in transfer reinforcement learning (including varying transfer beliefs) and exploring multi-robot collision avoidance problems mixed with more complex fixed and moving obstacles.

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

References

1. Bojarski, M. et al. End to end learning for self-driving cars. arXiv: 1604.07316 [cs.LG], 2016

2. Casanova, D., Tardioli, C., & Lemaître, A. (2014). Space debris collision avoidance using a three-filter sequence. Monthly Notices of the Royal Astronomical Society, 442(4), 3235-3242.
3. Chen, J. X. (2016). The Evolution of Computing: AlphaGo. Computing in Science & Engineering, 18(4), 4-7.
4. Churchland, P. S., & Sejnowski, T. J. (2016). The computational brain. MIT press.
5. Coates, A., B. Huval, T. Wang, D. Wu, A. Ng. Deep learning with COTS HPC systems. International Conference on Machine Learning, 2013
6. Chen, Y.F., M. Liu, M. Everett, and J. P. How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. arXiv: 1609.07845 [cs.MA] 2016
7. Dean, J. et al. Large scale distributed deep networks. International Conference on Neural Information Processing Systems. Curran Associates Inc., 2012
8. Dieleman, S., B. Schrauwen. End-to-end learning for music audio. IEEE International Conference on Acoustics, Speech and Signal processing. IEEE, 2014
9. Ding, Z., N. Nasrabadi, Y. Fu. Task-driven deep transfer learning for image classification. IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2016
10. Fahimi, F., C. Nataraj, H. Ashrafiuon. Real-time obstacle avoidance for multiple mobile robots. Robotica, 2009
11. Fernandez, F., M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. International Joint Conference on Autonomous Agents and Multiagent Systems (Vol. 58, pp.720-727). 2006
12. Frommberger, L. (2008). Learning to behave in space: A qualitative spatial representation for robot navigation with reinforcement learning. *International Journal on Artificial Intelligence Tools*, *17*(03), 465-482.
13. Fujii, T., Arai, Y., Asama, H., & Endo, I. (1998, May). Multilayered reinforcement learning for complicated collision avoidance problems. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on* (Vol. 3, pp. 2186-2191). IEEE.
14. Goerlandt, F., & Kujala, P. (2014). On the reliability and validity of ship–ship collision risk analysis in light of different perspectives on risk. Safety science, 62, 348-365.
15. Hinton, G., L Deng, D Yu, GE Dahl, A Mohamed, N Jaitly, A Senior, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Processing Magazine 29 (6), 82-97, 2012.

16. Hinton, G., O. Vinyals, and J. Dean, Distilling the Knowledge in a Neural Network. arXiv: 1503.02531v1 [stat.ML] 9 Mar -2015
17. Hourtash, A. M., Hingwe, P., Schena, B. M., & Devengenzo, R. L. (2016). U.S. Patent No. 9,492,235. Washington, DC: U.S. Patent and Trademark Office.
18. Keller, J., D. Thakur, J. Gallier, V. Kumar. Obstacle avoidance and path intersection validation for UAS: A B-spline approach. International Conference on Unmanned Aircraft Systems0 (pp.420-429), IEEE, 2016
19. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. International Journal of Robotics Research, 5(1), 1986
20. Krizhevsky, A., I. Sutskever, G. Hinton. ImageNet classification with deep convolutional neural networks. Communications of the Acm, 60(2), 2012
21. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
22. Liu, X. and Jin, Y. (2018) Transfer Reinforcement Learning: Task Similarities and Transfer Strategies. In preparation. 2018.
23. Machado, T., Malheiro, T., Monteiro, S., Erlhagen, W., & Bicho, E. (2016, May). Multi-constrained joint transportation tasks by teams of autonomous mobile robots using a dynamical systems approach. In Robotics and Automation (ICRA), 2016 IEEE International Conference on (pp. 3111-3117). IEEE.
24. March, J. G. (1991). Exploration and exploitation in organizational learning. Organization science, 2(1), 71-87.
25. Mastellone, S., D. Stipanovic, C. Graunke, K. Intlekofer, M. Spong. Formation control and collision avoidance for multi-agent non-holonomic systems: theory and experiments. The International Journal of Robotics Research, 2008, Vol.27(1), pp.107-126
26. Matarić, M. J. (1997). Reinforcement learning in the multi-robot domain. In *Robot colonies* (pp. 73-83). Springer US.
27. Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller. Playing Atari with deep reinforcement learning. arXiv:1312.5602v1 [cs.LG], 2013
28. Mukhtar, A., Xia, L., & Tang, T. B. (2015). Vehicle detection techniques for collision avoidance systems: A review. IEEE Transactions on Intelligent Transportation Systems, 16(5), 2318-2338.
29. Ohn-Bar, E., & Trivedi, M. M. (2016). Looking at humans in the age of self-driving and highly automated vehicles. IEEE Transactions on Intelligent Vehicles, 1(1), 90-104.
30. Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.

31. Parisotto, E., Jimmy Lei Ba, Ruslan Salakhutdinov, Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning, arXiv:1511.06342v4 [cs.LG]
32. Shiomi, M., Zanlungo, F., Hayashi, K., & Kanda, T. (2014). Towards a socially acceptable collision avoidance for a mobile robot navigating among pedestrians using a pedestrian model. International Journal of Social Robotics, 6(3), 443-455.
33. Silver, D. et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529(7587):484.
34. Tang S., V. Kumar. A complete algorithm for generating safe trajectories for multi-robot teams. International Symposium on Robotics Research, 2015
35. Taylor, M., P. Stone. Cross-domain transfer for reinforcement learning. International Conference on Machine Learning. ACM, 2007
36. Torrey, L., J. Shavlik, T. Walker, R. Maclin. Skill acquisition via transfer learning and advice taking. European Conference on Machine Learning. Springer Berlin Heidelberg, 2006
37. van Hasselt, H., Guez, A, Silver, D. Deep Reinforcement Learning with Double Q-learning, arXiv:1509.06461v3 [cs.LG], 2015
38. Wang, F. Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y., Dai, X., ... & Yang, L. (2016). Where does AlphaGo go: from Church-Turing thesis to AlphaGo thesis and beyond. IEEE/CAA Journal of Automatica Sinica, 3(2), 113-120.
39. Wang, Z, School T., Hessel, M. van Haselt, H. Lanctot, M. de Freitas, N. Dueling Network Architectures for Deep Reinforcement Learning, arXiv:1511.06581v3 [cs.LG] 5 Apr 2016b
40. Watkins, C. Learning from delayed rewards, 1989. Doctoral dissertation, University of Cambridge.
41. Yu, A., R. Palefsky-Smith, R. Bedi. Deep reinforcement learning for simulated autonomous vehicle control. 2016
42. Zou, X., Alexander, R., & McDermid, J. (2016, June). On the Validation of a UAV Collision Avoidance System Developed by Model-Based Optimization: Challenges and a Tentative Partial Solution. In Dependable Systems and Networks Workshop, 2016 46th Annual IEEE/IFIP International Conference on (pp. 192-199). IEEE.