DYNAMIC ROUTING FOR NAVIGATION IN CHANGING UNKNOWN MAPS USING DEEP REINFORCEMENT LEARNING

Yuci Han[a], Alper Yilmaz[a]

[a] Photogrammetric Computer Vision Laboratory,
The Ohio State University, Columbus, OH 43210, USA
{han.1489, yilmaz.15}@osu.edu

Commission IV, WG 5ICWG

KEY WORDS: Navigation, Dynamic Routing, Reinforcement Learning, Visual Odometry

ABSTRACT:

In this work, we propose an approach for an autonomous agent that learns to navigate in an unknown map in a real-world environment. Recognizing that the real-world environment is changing overtime such as road-closure happening due to construction work, a key contribution of our paper is adopt the dynamic adaptation characteristic of the reinforcement learning approach and develop a dynamic routing ability for our agent. Our method is based on the Q-learning algorithm and modifies it into a double-critic Q-learning model (DCQN) that only uses visual input without other aids such as GPS. Our treatment of the problem enables the agent to learn the navigation policy while interacting with the environment. We demonstrate that the agent can learn navigating to the destination kilometers away from the starting point in a real world scenario and has the ability to respond to environment changes while learning to adjust the routing plan dynamically by adjusting the old knowledge. The supplementary video can be accessed at the following link: https://www.youtube.com/watch?v=tknsxVuNwkg.

1. INTRODUCTION

With the introduction of deep reinforcement learning (DRL) in recent years, robotics field has started to widely use it. For robot navigation problem, the majority of research demonstrate that DRL agents can learn to navigate in confined and known virtual indoor environments such as AI2THOR and Gibson (Zhu et al., 2016) (Xia et al., 2018), maze (Mirowski et al., 2016) and 3D games (Lample and Chaplot, 2016). In contrast to the virtual environment, navigation in the unknown real-world environment has following characteristics: (1) constantly changing environment (2) complicated scenarios (3) irregular trajectories (4) large scale setups and (5) unknown map. Hence, training an agent for real world navigation problem is computationally more demanding. In this work, we propose a reinforcement learning method with replay buffer to store all the learned environment information and enable the agent to extract experience from the buffer to accelerate the learning progress in the case environment dynamically changes.

We investigate two tasks in this paper: (1) long-range real-world navigation in an unknown map; (2) dynamic rerouting due to changes in the environment. For the navigation task, we train the agent to traverse to a target destination using only visual observations similar to that of a human that can learn to find the destination by only observing immediate surroundings. The agent exploits an *action policy* that rewards reaching the destination while interacting with the environment. Compared with most state-of-art research, our primary contributions is to apply DRL approach to a real-world dynamic routing navigation problem in an unknown map. We show that the agent can learn new skills without forgetting the old knowledge while adapting to the changes in the environment which is considered a dynamic learning process. To evaluate the feasibility of the dynamic routing task, we modify the map by closing access to roads and let the agent relearn to reach to the destination based on the experience.

Training an agent in a DRL framework requires an interactive environment which is not provided in typical dataset collection techniques. To this end, we made an environment based on (Zhu et al., 2016) approach that enables the agent to freely move and rotate to collect observations of various real street scenes.

We propose DCQN method for these tasks. Our experiments demonstrate that the agent can swiftly learn the navigation task and reroute without knowing the map. We also compare DCQN with target-driven visual navigation algorithm (Zhu et al., 2016) and show that DCQN is significantly faster than the target-driven actor-critic algorithm.

2. RELATED WORK

In this section, we provide a brief overview of related research in the areas of DRL and navigation using DRL.

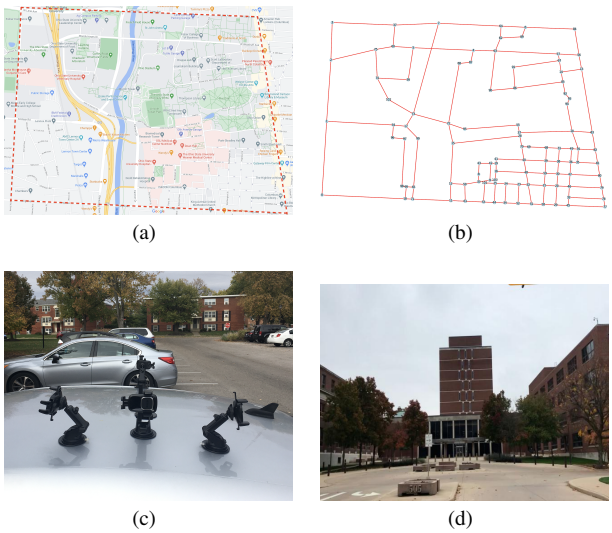
Deep reinforcement learning. DRL consists of model-free RL and model-based RL. For model-free RL, (Mnih et al., 2013) propose the most famous Q-Learning algorithm which has been demosntrated to successfully surpass human performance in an Atari game. (van Hasselt et al., 2015) address the Q value overestimation problem with double Q-Learning network. Additionally, C51 (Bellemare et al., 2017), and QR-DQN (Dabney et al., 2017) provide alternative value based RL techniques. Model-free RL includes a policy models such as A2C/A3C (Mnih et al., 2016), PPO (Schulman et al., 2017) and TRPO (Schulman et al., 2015). DDPG (Lillicrap et al., 2019), TD3 (Fujimoto et al., 2018), SAC (Haarnoja et al., 2018) that are considered advanced techniques based on both policy optimization and Q-Learning.

DRL navigation. Recently, navigation research based on RL have shown some promising results that can train agents in

simulated small indoor environments for short trajectories (Zhu et al., 2016). (Chang et al., 2017) , (Anderson et al., 2017), (Bruce et al., 2017), (Mo et al., 2018). The inherent problem of these simulated virtual environments (Dosovitskiy et al., 2017), (Kolve et al., 2017), (Shah et al., 2017), (Wu et al., 2018) is that the scenes tend to lack diversity and are visually simple. In our work, we use real world environment that consists of vegetation, moving vehicles and buildings which is diverse and real.

3. ENVIRONMENT

In this section, we introduce the CampusNav environment we have generated for the navigation and dynamic routing task. The environment is necessary for repeatability and testing of the algorithm. We should note that the algorithm can run when the agent is placed in an new unknown area. CampusNav spans the Ohio State University campus and surrounding region with a total area of 4.5 km^2 (see Fig. 1(a)). While constructing the map graph (see Fig. 1(b)), we extracted road network as a graph composed of edges and nodes at the intersections. We note that similar data can also be obtained from google street side images. The CampusNav dataset is available at: https://github.com/superhan2611/dynamic-routing.



(a) (b) (c) (d)

Figure 1. CampusNav covers a range of 2.5km x 1.7km, we illustrate the road network as an undirected graph. The image data is collected by iPhone placed on a moving car.

The CampusNav environment is built based on the approach proposed by (Zhu et al., 2016) and contains the following information:

Location. The locations are the coordinates of the intersection positions. We select 101 intersections in the test area to collect image data. These locations form the nodes of an undirected graph covering an area of 2.5km x 1.7km. The agent takes observations and performs action at these locations.

Observation. The observations are the RGB images collected by iPhones placed on the top of the moving vehicle (see Fig. 1(c)). The agent observes all four orientations, north, south, east and west (see Fig. 2) to decide on an action. The raw image resolution is set at 640×480. We use the ImageNet pretrained ResNet-50 (He et al., 2015) to produce 2048-d features as the input of state in the training process.

State-action transition graph. Reinforcement learning requires the agent to interact with the environment. This interactive process contains a sequence of actions and corresponding state observations $(s_1, a_1, ..., a_{N-1}, s_N)$. To this end, the state-action transition graph are used to store the state-action transition information of the environment. Although the graph is used to construct the environment, the agent only sees the acquired image at the particular location without knowing other information in the navigation process.

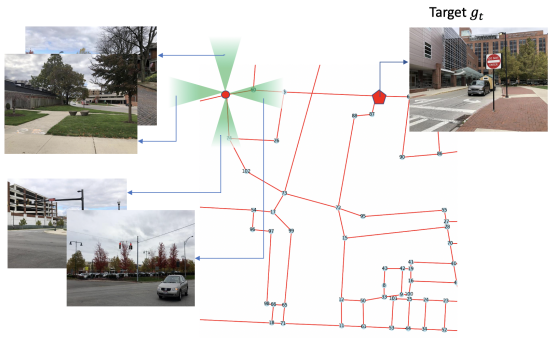


Figure 2. Discrete locations and agent's observations at each position

4. METHODOLOGY

This section formulates the problem and specifies the algorithm architecture for these tasks required in an RL framework.

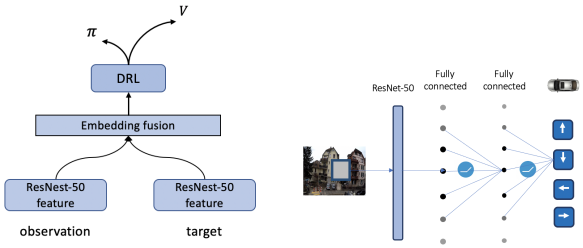
4.1 Problem Formulation

The goal of this paper is to teach the autonomous agent the skill to navigate to the destination starting from arbitrary location with the minimum action steps using the images of the scene as vehicle moves. Additionally, the agent is taught the ability to dynamically reroute depending on the changes to the road network. In our model, we take the image that the agent sees as the state input. The target scene is specified by another image. The output is the action that the agent takes such as move forward, move back, turn right and turn left depending on the policy π learned by the agent.

We formulate the learning problem as a Markov Decision Process (MDP) and implement the RL framework. The key elements of RL are state, action space and task reward. The agent uses visual odometry and has two inputs: the ResNet-50 features extracted from the images from the agent's current state s_t and the target scene g_t. To find the spatial arrangement between the current and target locations, we project them into the same embedding space represented as embedding fusion input in Fig. 3(a), so that the view-geometric relations are preserved and a joint representation of current state and the target scene are formed and used as the input of the DRL algorithm.

The navigation task that takes one of the four actions is performed on an irregular grid unlike an indoor environment (see Fig. 1(b)). Hence, we define a range of directional representations for the actions. In particular, move forward represents moving in any direction between -22.5° and 22.5°, moving backward between 67.5° and -67.5°, turning right between 22.5° and 67.5°, and finally turning left between -22.5° and -67.5°.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-1-2021-145-2021 | © Author(s) 2021. CC BY 4.0 License.



(a) (b)

Figure 3. (a) The outline of the network architecture, the ResNet-50 are pre-trained on ImageNet. (b) Schematic illustration of the critic network. The ResNet-50 extracts feature from the original image.

For the reward design, since our goal is to minimize the action sequence length to navigate to the target, we consider three situations: reward for reaching the goal (1.0) upon task completion, time penalty (-0.005) to encourage shorter trajectory and collision (-0.01) to avoid hitting objects.

4.2 Double Critic Deep Q-Learning Model

Arguably, the navigation task can be considered as a Markov Decision Process, with state space $\mathcal{S}$ and action space $\mathcal{A}$. In our problem, the environment is an unknown geographic area and the state space $\mathcal{S}$ is finite. We consider navigation task as an off-policy learning process and let the autonomous agent learn the policy based on experience instead of exploring the environment for collecting experience, which is a less efficient and time consuming task. To this end, we store the past traversed environment information: state, action, next state and immediate reward as the agent's experience $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time-step t in a replay buffer and implement the deep Q-Learning algorithm. The drawback of traditional Q-Learning is that it overestimates action values under certain conditions. Therefore, we introduce a trivial modification and propose a DCQN model to reduce over estimations. Similar to the Double Q Learning approach (van Hasselt et al., 2015), it is the progress towards finding general solutions rather than a deterministic sequence of steps that would be less robust.

To achieve this, we build two pairs of target-critic networks. Similar to the usual Deep Q Learning, the target networks share the same parameter with the corresponding critic network and are only periodically updated. Under a given policy, the critic (Q) function with action a at state s

$$Q_\pi(s,a) = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... | s_t = s, a_t = a, \pi] \quad (1)$$

is parameterized using the two full-connected layers of the perceptron shown in Fig. 3(b).

The goal of the agent is to select the action that maximizes the critic function at each time step t after making an observation(s). This is followed by taking the optimum action(a). In each learning episode, we draw samples of experience (s, a, r, s') from the replay buffer at random drawn from a uniform distribution. The critic network generates the Q value of the policy $\pi = P(a|s)$. The theoretical Q value should be $r + \gamma \max_{a'} Q'(s', a')$ and $\max_{a'} Q'(s', a')$ is calculated by the target network. In this paper, we propose two target and critic networks and choose the minimum value generated by these two networks as the target Q' value to avoid overestimating problem. The Q-Learning updates the network parameters using the temporal-difference with the loss function given as follows:

$$L_i(\theta i) = E_{(s,a,r,s')}[(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2] \quad (2)$$

The two critic networks are updated according to the TD error. After a certain number of episodes, the target network is also updated using the parameters of corresponding critic network. The learning structure is shown in Fig. 4.

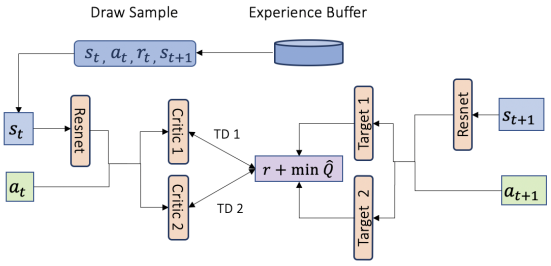


Figure 4. Schematic illustration of Double Critic Deep Q learning model.

5. EXPERIMENTS

In this section, we implement the DCQN model for the navigation and dynamic routing tasks. We first show our agent's capability to navigate in a real campus environment and then demonstrate its performance as it responds to the changes in the environment. We also compare our model with the target-driven AC model (Zhu et al., 2016) on the navigation task in AI2-THOR simulated indoor environment.

5.1 Experimental setup

We randomly initialize the weights of the two critical networks. The input is the ResNet-50 extracted features from 640x480 raw image. All experiments are conduct on a NVIDIA Titan V using Pytorch with Adam optimizer. While training the critic networks, we use learning rate $\eta = 10^{-4}$ and discount rate $\gamma = 0.95$ in equation (1) and (2). The size of the replay buffer is 412.

5.2 General Navigation in CampusNav Environment

In our experiments performed in the CampusNav environment, we observe that the agent trained with the DCQN successfully learns the navigation task. For this task, we investigate two different agent architectures: single-target (singleNav) and multi-target (multiNav). The *single-target* agent is trained with one specific goal while the *multi-target* agent is trained with multiple goals simultaneously (see goal settings from Table 1). The training progress for both agents are shown in Fig. 5. In the rest of the discussion, the rewards in figures indicate the cumulative rewards computed at each episode as the agent reaches the destination. As shown in Fig. 5 the agent trained with single target converges faster and is more stable than the one trained with multiple targets. However, after certain number of episodes, the *multi-target* agent finds the same goal as the *single-target* agent and has stronger navigation ability as it is trained with multiple goals.

For testing repeatability, we rerun the experiment for 100 episodes and plot the number of steps the agent took to reach the

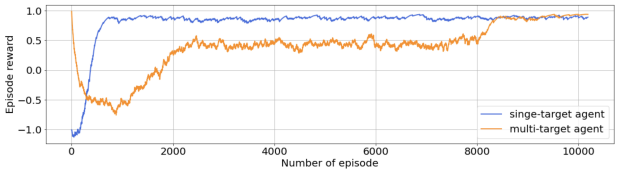
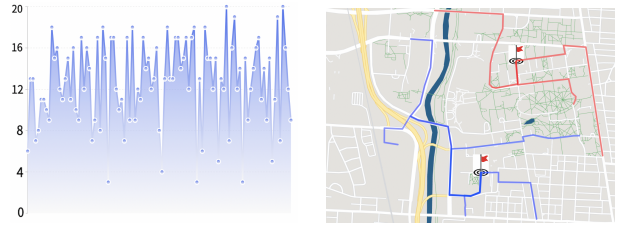


Figure 5. Training curves tracking the agent's episode reward.

singleNav	multiNav
Dreese Lab	Dreese Lab
	Town house
	Cinema
	Intramural fields

Table 1. Training goal settings

destination. Fig. 6(a) shows that the multi-target agent is able to find the target in less than 20 steps and covers all the nodes on the map which suggests that the agent could reach to the destination starting from any unknown location on the map. Within the same training episodes, we observe that there are some miss-trained points for single-target agent which leads to an unsuccessful navigation process as the agent only covers 95% of the nodes on the map. Fig. 6(b) visualizes the trajectories of the multi-target navigation with random starting point to 2 different goal destinations.



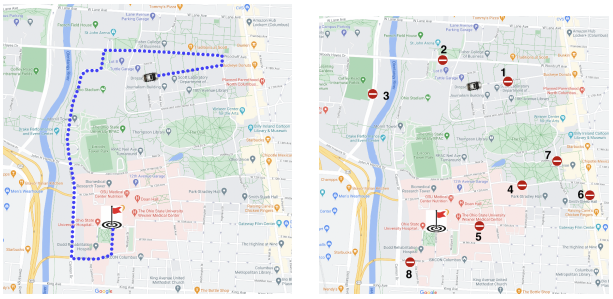
(a) (b)

Figure 6. Number of steps taken for 100 test episodes and the resulting trajectories generated for visualization for the multi-target agent.

5.3 Navigation in a Dynamically Changing Map

A critical experiment for our method is the dynamic routing task which demonstrates the proposed agent's ability to adapt to the continuously changing environment. The change in the environment are obtained by closing the roads for instance due to construction. This is represented by updating the state-action transition graph where value -1 indicates the unreachable to the next state. By definition, the dynamic routing task requires learning of the map and the agent is expected to update the policy in accordance with the changes. Rather than relearn the whole environment, we focus on constantly adapting the old agent skills to the changing map. To achieve this goal, our agent is firstly assigned a goal and trained for single-target navigation task (result shown in Fig. 7(a)). When the map changes due to road closure as illustrated in Fig. 7(b), the agent quickly relearn the policy based on the changes based on past experience.

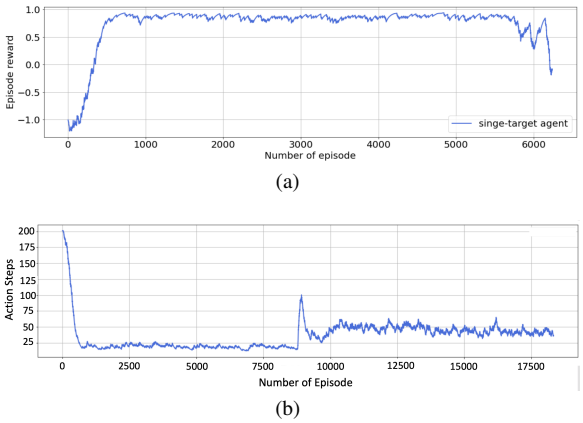
Figure 8(a) shows the training cumulative rewards earned in each episode. The performance is stable and the change of the map doesn't cause instabilities during training. Each time a road is closed, the agent is able to update its policy in less than 3 minutes under 300 episodes. In real application, this indicates that the car could stop and a new solution will be found in few



(a) (b)

Figure 7. (a) The navigation route learned for a single-target agent. The vehicle and the flag represent the starting point and the target correspondingly. (b) The stop sign represents road closure and follows the number order.

minutes. We can see from the process in Fig. 9(a)-9(h), the light blue trajectory represents the original route while the dark red trajectory shows the agent's adaptive rerouted path when the map changes.



(a)

(b)

Figure 8. (a) Training curves tracking the agent's episode reward while closing one road at a time. (b) Training curves tracking the agent's episode steps while closing roads simultaneously.

In another experiment, we randomly close 11 roads simultaneously at around 8000th episode which can be seen as the pulse in the plot in Fig. 8(b). After this closure, it is observed that the learning performance of the agent stabilizes in less than 500 episodes which demonstrates a significantly fast and successful dynamic learning ability. Since the map becomes complicated after this significant number of closures, the number of steps taken to reach the goal in one episode is more than it was before. Figures 9(i) and 9(l) illustrate examples of the adjusted route plan compared with former policy.

5.4 Comparative analysis

We introduce a new model, which we refer to as the Double Critic Q-Learning Network for navigation in unknown and dynamically changing maps. To compare the performance of our DCQN algorithm against a baseline target-driven Actor-Critic (AC) model given in (Zhu et al., 2016), we conduct experiments using the AI2-THOR framework. AI2-THOR is a simulated indoor environment which includes a set of interactive scenes and provides accurate modeling of the world physics.

We train the model on 14 different target scenes selected from two sets of indoor environment: bathroom and bedroom. For

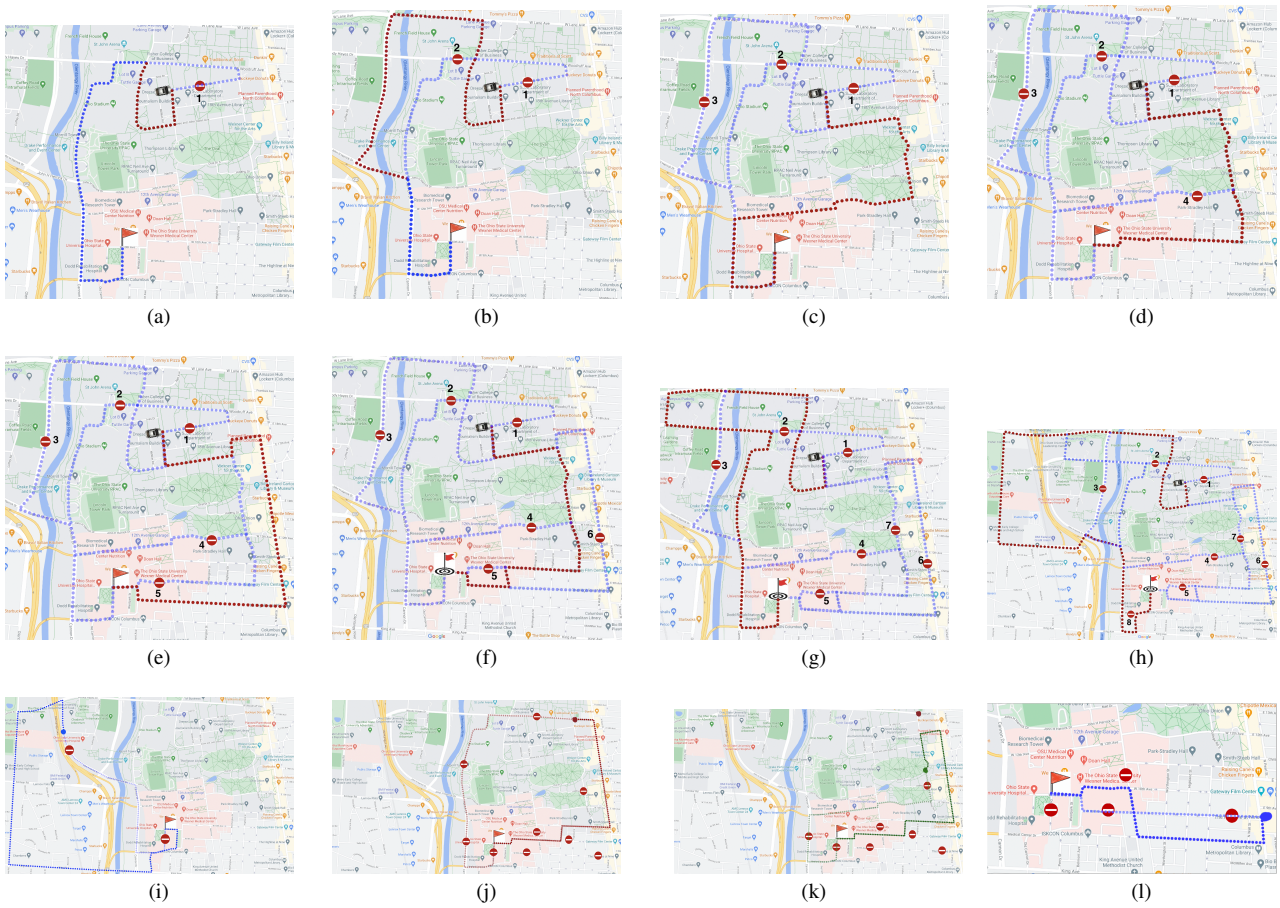


Figure 9. Agent's dynamic routing plan regarding to the change of the map

evaluation, we do the navigation task for 100 episodes and set a maximum number of steps to 200 which means the agent will give up finding the destination in a particular episode if it can't reach it in 200 steps. Throughout the experiment, for a just comparison of learning efficiencies of the two methods, we limit the training time to 5 minutes for each target and then evaluate the learning performance.

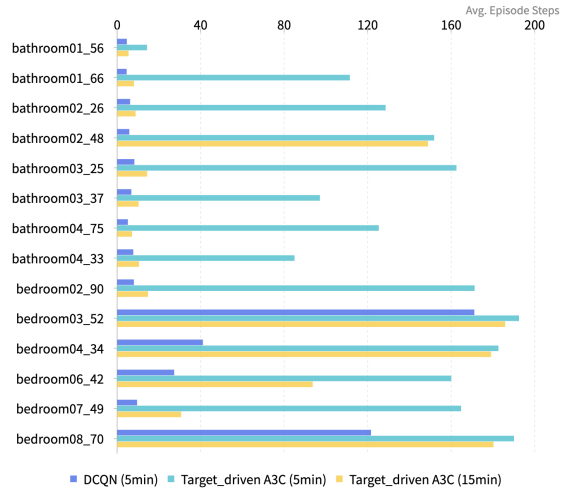


Figure 10. Comparison of the DCQN agent with Target-driven AC agent

Figure 10 provides a comparison of the two models and shows that the DCQN method outperforms the Target-driven model on all 14 target navigation tasks. DCQN agent converges faster and is more stable than the AC agent of (Zhu et al., 2016). With the same training time, the average step length of AC agent is 4.5 times more than the DCQN agent. We also trained the AC agent for an additional 15 more minutes, and show in Fig. 10 that the AC agent improves its performance , yet there is still a big gap in performance when compared to DCQN model. One possible reason for surpassing the AC model is we build the replay buffer which stores the (s_t, a_t, s_{t+1}, r) pairs based on the state-action transition graph and train the network according to TD error which is more efficient than exploring the environment from the very beginning in AC model.

6. CONCLUSION

We introduced the reinforcement learning approach for autonomous navigation of an agent that can dynamically adapt itself in changing real-world environments. The agent uses visual odometry to generate relative trajectory and navigates using a MDP process and a dynamic learning characteristic of DRL approach that allows the agent to adjust the navigation policy as the map changes. Our approach can easily be generalized to more complicated and continuously changing environments and maps. We demonstrated the navigation performance of our approach in large scale real-world environments with multiple destinations where the environment dynamically changes. We also showed that the DCQN algorithm outperforms the Target-driven AC model in commonly used AI2-THOR environment for indoor navigation problem.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-1-2021-145-2021 | © Author(s) 2021. CC BY 4.0 License.

The limitations of our approach are: (1) Our RL algorithm is model based and is trained in the certain environment, therefore, it is lack of generalization to new environment or unseen target. (2) We formulate the map as a graph and perform the RL algorithm at the nodes of the graph with four actions. As for more complex map, the number of node and action space will be much larger. Training agent in such an environment might be time consuming. Future work will involve generalizing the navigation ability for unseen environment or target and improving the learning efficiency for more complicated map.

REFERENCES

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I. D., Gould, S., van den Hengel, A., 2017. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. *CoRR*, abs/1711.07280.

Bellemare, M. G., Dabney, W., Munos, R., 2017. A Distributional Perspective on Reinforcement Learning. *CoRR*, abs/1707.06887.

Bruce, J., Sünderhauf, N., Mirowski, P., Hadsell, R., Milford, M., 2017. One-Shot Reinforcement Learning for Robot Navigation with Interactive Replay. *CoRR*, abs/1711.10137.

Chang, A. X., Dai, A., Funkhouser, T. A., Halber, M., Nießner, M., Savva, M., Song, S., Zeng, A., Zhang, Y., 2017. Matterport3D: Learning from RGB-D Data in Indoor Environments. *CoRR*, abs/1709.06158.

Dabney, W., Rowland, M., Bellemare, M. G., Munos, R., 2017. Distributional Reinforcement Learning with Quantile Regression. *CoRR*, abs/1710.10044.

Dosovitskiy, A., Ros, G., Codevilla, F., López, A. M., Koltun, V., 2017. CARLA: An Open Urban Driving Simulator. *CoRR*, abs/1711.03938.

Fujimoto, S., van Hoof, H., Meger, D., 2018. Addressing Function Approximation Error in Actor-Critic Methods. *CoRR*, abs/1802.09477.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *CoRR*, abs/1801.01290.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385.

Kolve, E., Mottaghi, R., Gordon, D., Zhu, Y., Gupta, A., Farhadi, A., 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. *CoRR*, abs/1712.05474.

Lample, G., Chaplot, D. S., 2016. Playing FPS Games with Deep Reinforcement Learning. *CoRR*, abs/1609.05521.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2019. Continuous control with deep reinforcement learning.

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A. J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., Hadsell, R., 2016. Learning to Navigate in Complex Environments. *CoRR*, abs/1611.03673.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., Kavukcuoglu, K., 2016. Asynchronous Methods for Deep Reinforcement Learning. *CoRR*, abs/1602.01783.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. A., 2013. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602.

Mo, K., Li, H., Lin, Z., Lee, J., 2018. The AdobeIndoorNav Dataset: Towards Deep Reinforcement Learning based Real-world Indoor Robot Visual Navigation. *CoRR*, abs/1802.08824.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., Abbeel, P., 2015. Trust Region Policy Optimization. *CoRR*, abs/1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Shah, S., Dey, D., Lovett, C., Kapoor, A., 2017. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *CoRR*, abs/1705.05065.

van Hasselt, H., Guez, A., Silver, D., 2015. Deep Reinforcement Learning with Double Q-learning. *CoRR*, abs/1509.06461.

Wu, Y., Wu, Y., Gkioxari, G., Tian, Y., 2018. Building Generalizable Agents with a Realistic and Rich 3D Environment. *CoRR*, abs/1801.02209.

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., Savarese, S., 2018. Gibson Env: Real-World Perception for Embodied Agents. *CoRR*, abs/1808.10654. http://arxiv.org/abs/1808.10654.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., Farhadi, A., 2016. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. *CoRR*, abs/1609.05143.