Reinforcement Learning for Active Monitoring of Moving Equipment in 360-Degree Videos

Nath N.D., Cheng C., Behzadan A.H.
Texas A&M University, USA
abehzadan@tamu.edu

Abstract. Computer vision techniques have been introduced recently to assist with visual surveillance of jobsite activities. However, multiple reality capture devices are needed to guarantee uninterrupted view of key objects. We propose to use deep learning with reinforcement learning (RL) to create a self-navigating active vision camera. The trained RL camera gains sufficient spatiotemporal knowledge to fix its gaze on an object by dynamically adjusting its position and view angle. We use a deep Q-learning network (DQN) to decide whether to move or rotate the camera to monitor moving forklifts in a 360-degree video. Results show that the RL camera can find a new position and angle with better view of the forklift in 73% of cases, and in the remaining 27% of cases, the visibility of the forklift remains unchanged. This indicates the effectiveness of the RL agent in locating the object of interest in complex and dynamic real-world settings.

1. Introduction

In recent years, the number of photos and videos collected daily by various digital capture devices has exponentially increased. In 2020 alone, 1.43 trillion digital images were captured with more than 91% taken on mobile phones (Canning, 2020). This growth in the volume of visual data can be in part attributed to the ubiquity of mobile devices (e.g., smartphones, tablets), digital cameras, and unmanned aerial vehicles (UAVs) also known as drones with onboard cameras. Visual data is also widely utilized to record construction fieldwork and commonly used to document progress reports, complement requests for information (RFIs), prepare safety training materials, and litigate claims. Moreover, continuous and unobtrusive monitoring of jobsite activities is key to work progress measurement and monitoring safety compliance on construction sites. Advancements in computer vision (CV) and artificial intelligence (AI) have created new solutions to automate visual surveillance tasks. For example, Nath and Behzadan (2020a) applied deep learning (DL) to detect common construction objects (e.g., building, equipment, worker) in real-time under diverse visual conditions. Nath et al. (2020b) and Fang et al. (2018) proposed DL techniques to monitor workers' compliance with regulations pertaining to the use of personal protective equipment (PPE) (i.e., hard hat, safety vest).

While the uptake of AI integration into construction practices is expected to continue, the next-generation construction and manufacturing systems will enable humans and AI to collaborate in more meaningful ways (NSF, 2020, Autodesk, 2020), as evident by the success of similar efforts in other domains such as medicine (McCoy et al., 2020, Tschandl et al., 2020), data science (Wang et al., 2019), and business management (Sowa et al., 2021). A key prerequisite to the successful completion of complex machine-operated tasks with vision-based AI is the ability of the machine to understand the content and context of its surroundings. In a nutshell, it is critical for an AI-enabled machine to not only know what objects are located in its proximity, but also infer the spatiotemporal relationships between those objects. This crucial ability, however, may be hindered for several reasons. In a constantly evolving workspace such as a construction site, for example, objects are often on the move and frequently occlude one another. In fact, occlusion is a major impediment to the performance of CV algorithms (Hoiem

https://doi.org/10.7146/aul.455.c204

et al., 2011). A previous study, for instance, has found a significant drop in the performance of facial recognition systems when the face is partially occluded (Ekenel and Stiefelhagen, 2009). Similarly, when an AI algorithm is used by a fixed-viewpoint camera to monitor the workspace for safety compliance or progress evaluation, the visual recognition task may yield low accuracy because of workers and equipment not being fully visible due to displacement or occlusion. A rhetorical solution to this problem is to move the camera (i.e., adjust the viewpoint) or install multiple cameras so that objects of interest always appear sufficiently visible. However, due to the constantly evolving nature of construction sites and cluttered workspaces, and the need for continuous calibration and installation, this approach will translate into a daunting task, consuming a significant amount of project resources.

To address the abovementioned challenge, this study proposes to design an active vision camera capable of navigating in the scene and operating autonomously while searching for the optimal viewpoint from which occlusion-free scenes can be captured for a given task. To validate the performance of the designed methods, an AI-enabled drone camera (with 2 degrees of freedom) is trained with reinforcement learning (RL) and tested in an active work environment.

2. Literature review

DL is a subset of machine learning (ML) and is characterized as a neural network with more than three layers (Chollet, 2021). Deep learning models have been successfully trained to perform tasks in various fields including but not limited to visual recognition (Krizhevsky et al., 2017), natural language processing (Deng and Liu, 2018), and self-driving vehicles (Rao and Frtunikj, 2018). RL, on the other hand, expands the traditional boundaries of ML by dynamically training a function to take optimal actions given the environment states based on continuous feedback to maximize its cumulative reward. RL was first tossed in the early 1980s in psychology, to describe a trial-and-error methodology where animals learn and change specific behaviors based on their perceived experiences (Busoniu et al., 2010). Similarly, modern RL algorithms rely on trial-and-error to train computational agents to make decisions. In the late 1990s, the use of RL was expanded to other domains including robotics (Kormushev et al., 2013), game playing (Szita, 2012), and industrial process control (Nian et al., 2020).

In an RL problem, the learner is referred to as agent, and everything in the surrounding, that the agent can interact with, is called the environment (Chollet, 2021; Géron, 2019; Sutton and Barto, 2018; Szepesvári, 2010). At any given time, the part of the environment that is accessible to the agent can be described in a mathematical form that is called the state of the environment. The RL agent can perform an action to transition from the current state to the next state of the environment. When an agent performs an action, it also receives feedback from the environment indicative of the merit of the action in meeting the ultimate learning goal of the problem. This feedback is described in a numerical form and referred to as reward (Chollet, 2021; Géron, 2019; Sutton and Barto, 2018; Szepesvári, 2010).

With advancements in processing speed and computing power, conventional RL algorithms and neural network architectures have been combined to form new Deep RL paradigms (Henderson et al., 2018). For example, the AlphaGo algorithm, trained with RL and deep neural networks, can win a game against a human player almost 100% of the time (Silver et al., 2016; Silver et al., 2017). Other important Deep RL applications include autonomous driving, industry automation, and natural language processing (NLP) (Zhang et al., 2018; Nguyen et al., 2020). In the civil engineering domain, RL and Deep RL have been primarily leveraged in transportation applications including traffic signal control (Lin et al., 2018; Muresan et al., 2019; Tan et al., 2019; Rasheed et al., 2020; Xu et al., 2020) and speed limit control (Wu et al.,

2018). For example, Wu et al. (2018) proposed a Deep RL-based model for dynamic control of posted speed limits in response to prevailing traffic. They defined variable speed limit (VSL) controllers as RL agents and trained them with the deep deterministic policy gradient (DDPG) algorithm. For this particular application, the state variables of the RL environment were the occupancy rate (reported by loop detectors) of each merge, upstream, and on-ramp lane, while the reward function was based on total travel time, crash probability, and vehicular emission. Compared to a baseline scenario where no speed control was used, trained agents were able to dynamically change vehicle speed limits in each traffic lane based on the traffic flow. Results indicated a reduced average travel time (ATT), by nearly 40 seconds, and fewer vehicle emergency braking by approximately 6%.

In another study, Muresan et al. (2019) trained and validated a Deep Q-learning Network (DQN) to control the duration of traffic lights. In a nutshell, DQN refers to the use of Q-learning algorithm for Deep RL model training (Li, 2017). Since such networks can leverage the advantages of deep convolutional neural networks (CNN), they are widely used in CV applications (Li, 2017, Jang et al., 2019). Using a simulated intersection with psycho-physical cars, Muresan et al. (2019) proceeded to represent the state space by a bit-level matrix that described the queue length, signal state, and time of the day. In each second, the RL agent took one of the two actions, namely continuing the current phase or commencing the next phase. Subsequently, the agent was rewarded or penalized based on the traffic discharge and waiting times of the vehicles. The DQN model, trained with 61 days of simulated data, reduced the average delay by 32% compared to an actuated controller, and by 37% compared to the fixed time control of traffic lights. Mullapudi et al. (2020) explored the use of DQN to enable a stormwater system to dynamically adapt its response to a storm by controlling distributed assets including valves, gates, and pumps. In a simulated storm environment, the state was described by water level and flow sensor readings. At any given time, considering the state, an RL agent took the action of opening a valve or turning on a pump. The agent was trained with three different reward functions. In the first function, the agent received a positive reward for maintaining the outflow below a specific threshold and a negative reward otherwise. In the second function, the agent was rewarded for reaching flows that were close to the desired flow threshold. Finally, in the third function, the agent received the highest reward for keeping the basin empty. It was found that the trained RL agent could immediately observe the impact of its control actions and make adjustments in a 4 km^2 simulated stormwater network. Yao et al. (2020) applied DQN to long-term pavement maintenance planning by representing the state of the environment using 42 features that described pavement condition. The action space was composed of three features, namely maintenance types, maintenance materials, and distress treatment, which considering different combinations, formed a total of 38 actions (i.e., types of treatment) including “do nothing”. Furthermore, the reward function was defined as the increase or decrease in cost effectiveness after taking each action. Using two case studies, the researchers demonstrated the ability of the RL agent to reduce maintenance cost and length of the pavement segment that had to be maintained, ultimately leading to better maintenance strategies that maximized the long-term cost effectiveness in a 15-years period. Lastly, dos Santos et al. (2013) proposed an RL-based method to control and transport structural components to designated locations and assemble truss-like 3D structures in a real-world constrained but dynamic environment. The state of the environment in this case was composed of the status of the components as well as the location of the robot. The RL agent was rewarded based on the assembly sequence and the selection of structural elements for each assembly point, with an error in assembly leading to a negative reward (i.e., penalty). The developed approach reduced the tedious task of developing an assembly plan by allowing robots to efficiently assemble and construct multiple 3D structures.

Despite past efforts in designing RL or Deep RL algorithms in civil engineering applications, successful use cases of these methods in the construction domain are yet to be documented. While the majority of previous work has devised simulated environments with controlled parameters, the efficacy of DL or Deep RL agents in real-world settings where environment parameters are subject to change remain largely unknown and needs to be investigated and benchmarked. In particular, jobsite safety and accident prevention is an active area of research with real implications (OSHA, 2019). Therefore, the goal of the research presented in this paper is to make new strides in enabling the application of Deep RL for safety monitoring in real-world settings. Particularly, an RL agent with active vision capability is trained to locate and monitor forklifts in a dynamic workspace with the ultimate goal of alerting workers of imminent contact collisions (e.g., due to the worker and forklift moving too close to each other, or the worker moving into the blind spot of the forklift). In addition to safety applications, uninterrupted monitoring of equipment can be of value to supply chain management, asset management, resource allocation, and productivity assessment.

3. Methodology

The environment. In this study, the environment is generated from a warehouse operation 360°-video (obtained from YouTube under Creative Commons license), which is hereinafter referred to as Pictor-360 video. As shown in Figure 1, each frame of this video encompasses a 360° panoramic view of the warehouse. The resolution of each frame is 5120×1080 from which a 420×420 window is cropped to mimic the camera view of an autonomous drone. At each time step, the drone can rotate 11.25° to left or right (i.e., rotation along the yaw axis) and 5° upward or downward (i.e., rotation along the pitch axis). These actions result in translating the 420×420 window by 160 pixels along the horizontal direction or by 60 pixels along the vertical direction, as shown in Figure 1. In total, there are 32 horizontal positions and 12 vertical positions for each of the 699 frames of the video, resulting in 32×12×699=268,416 discrete states of the RL environment.

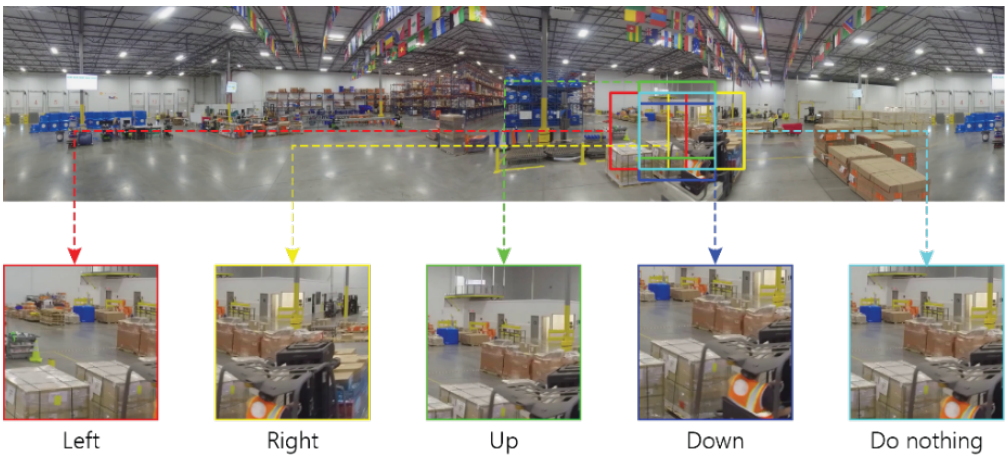


Figure 1: A sample frame of the Pictor-360 video and extracted camera views by taking different actions (source video courtesy of Direct Relief under CC BY license).

Q-learning. In each state, the RL agent can take one of the five actions of moving "up", "down", "left", "right", or "do nothing". The RL agent would take the action that will ultimately lead to the maximum reward. The value of the action in each state is called *action value* or *Q-value* (Géron, 2019, Szepesvári, 2010, Sutton and Barto, 2018), and mathematically denoted as

$\boldsymbol{Q(S_t, a)}$, where $\boldsymbol{S_t}$ is the state at time $\boldsymbol{t}$ and $\boldsymbol{a}$ is the action. Therefore, the agent learns to take the action with the maximum Q-value, i.e., $\boldsymbol{\mathrm{argmax}_a Q(S_t, a)}$. The term *Q-learning* refers to the algorithm for learning Q-values through iteration (Sutton and Barto, 2018). Equation (1) expresses the mathematical formulation for Q-learning, where $\boldsymbol{\alpha}$ is the learning rate, $\boldsymbol{r_t}$ is the reward received at time $\boldsymbol{t}$, and $\boldsymbol{\gamma}$ is the discount factor. Through this Equation, the model updates the Q-value for an action $\boldsymbol{a}$ at state $\boldsymbol{S_t}$, from $\boldsymbol{Q^{old}(S_t, a)}$ to $\boldsymbol{Q^{new}(S_t, a)}$. The value of reward $\boldsymbol{r_t}$ is defined based on the intersection over frame (IoF), which is defined as the percentage of the bounding box area of the forklift that intersects with the viewing frame of the RL camera.

$$Q^{new}(S_t, a) = Q^{old}(S_t, a) + \alpha[r_t + \gamma \max_a Q(S_{t+1}, a') - Q^{old}(S_t, a)] \quad (1)$$

3.1 Model Architecture

In this experiment, we propose to use a DQN model to predict the Q-values corresponding to the five actions named above (i.e., "up", "down", "left", "right", "do nothing"). For this purpose, the 420×420 RGB image viewed by the RL camera, is resized to a 224×224 resolution using bicubic interpolation (Shan et al., 2008). As shown in Figure 2, a VGG-16-based model is utilized to extract features. The choice of VGG-16 is due to its high performance in classifying numerous real-world objects (Simonyan and Zisserman, 2014). The VGG-16 network is amended with three convolution blocks, each consisting of a convolution with exponential linear unit (ELU) activation function, followed by a max-pooling layer. Features are then flattened and connected with a dense layer with 64 nodes. The final output layer consists of 5 nodes to output individual Q-values corresponding to the five actions.

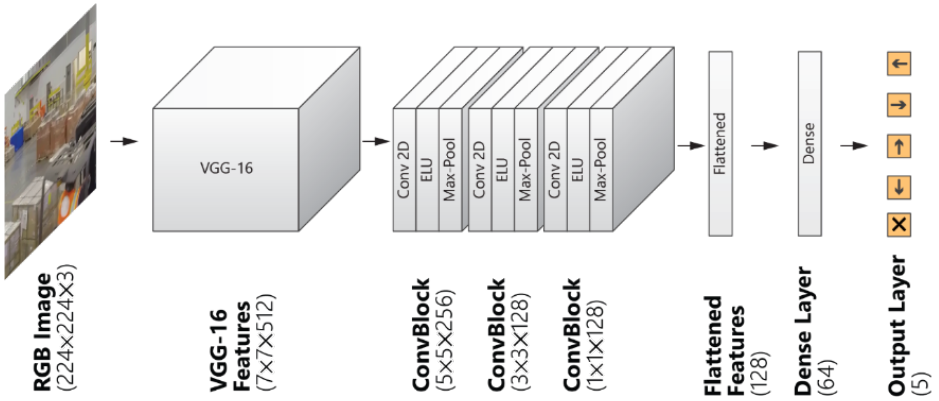


Figure 2: The architecture of the DQN model adopted for the Pictor-360 experiment.

3.2 Model Training

A well-known dilemma in training an RL agent is the *exploration vs. exploitation* problem. The goal of exploration is to allow the RL agent to take random actions to visit new states that were not previously traversed. However, exploitation describes a learning process where the RL agent takes the best action (a.k.a., greedy action) based on current knowledge to reinforce its learning (Géron, 2019, Sutton and Barto, 2018). In this paper, we adopt *ε-greedy policy* where at each state, the RL agent will explore a randomly selected action with probability ε ($0 \leq \varepsilon \leq 1$), and exploit this greedy action with probability $1 - \varepsilon$ (Géron, 2019; Sutton and Barto, 2018). Particularly, we start with a high ε ($\varepsilon_{\max} = 1.0$) to encourage the agent to explore. However, as the agent becomes mature through training, we linearly decay the value of ε to reach a minimum value ($\varepsilon_{\min} = 0.1$) so to focus the attention of the agent on exploiting its previously learned knowledge. The training stops at that minimum value to avoid overfitting.

In any given time step, the *experience* of the RL agent is composed of the following elements: current state, action taken, reward received, next state, and a Boolean value (i.e., true or false) indicating if the next state is the terminal state. To expedite training, in any one iteration, the agent is trained on multiple experiences (a.k.a., batch of experience) simultaneously rather than on a single experience (Schaul et al., 2015; Géron, 2019; Sutton and Barto, 2018). To train the model more effectively, we adopt a scheme called *experience replay* (Schaul et al., 2015; Géron, 2019; Sutton and Barto, 2018) where a fixed number of past experiences are stored in the RL agent's memory. In each iteration, several sample experiences are randomly drawn from the stored experiences and a sample batch of experience will be generated to train the RL agent. Particularly, we use a batch of size 200 and a deque data structure of 400,000 capacity to store the experiences. Also, we do not train the model during the first 2,000 iterations (a.k.a., *warm-up period*) and use these iterations only to gather enough experiences to fill up the deque memory. A complete list of the training hypermeters is presented in Table 1.

Table 1: Hyperparameters for the Pictor-360 experiment.

Category	**Hyperparameter**	**Value**
Environment	Number of states	268,416
	Number of actions	5
	Image resolution	420×420
	# Time steps in each episode	100
Reward	Step reward	IoF
	Discount factor, γ	0.995
Policy	$\varepsilon_{\max}$	1
	$\varepsilon_{\min}$	0.1
	$n_{\text{iteration}}$	10,000,000
Experience	Deque memory size	400,000
	Batch size	200
Training	Number of iterations	10,000,000
	Warm-up period	2,000 steps
	Target model update interval	2,000 steps
	Optimizer	Adam
	Learning rate	0.00001

4. Results and Discussion

Figure 3 shows the average reward received by the RL agent in 1,000-episode intervals. It can be seen that the model achieves higher rewards over time. During the test episode, the RL agent starts from 1,024 randomly selected positions to find better visibility of the forklift. The IoF at the very first frame is called initial IoF (I_O) while the maximum IoF achieved during one test episode is called maximum IoF (I_m). Figure 4(a) displays the I_O versus I_m. In this Figure, the 45° equality line represents the positions where $I_m = I_O$, i.e., the IoF of the forklift remains unchanged. In this experiment, all (100%) of the points are on or above the equality line (i.e., $I_m \geq I_O$), and 73% of the points are above the equality line (i.e., $I_m > I_O$). This finding implies that in 73% of the times the RL agent has found a position from where the forklift was more

visible, indicating the effectiveness of the RL agent in successfully locating the object of interest in complex and dynamic real-world settings. It is worth noting that in the remaining 27% of cases, the visibility of the forklift was preserved at the same level, and in no cases, the movement of the RL agent resulted in lower visibility. Figure 4(b) exhibits the percentage of times that the RL agent was able to find a better position from different initial IoFs. The Figure shows that when the initial IoF is low (e.g., 0.0-0.1), the RL agent can improve visibility in 65% of the times. However, if the initial IoF is higher (e.g., > 0.1), the RL agent performs better in improving visibility (i.e., 80% of the times or more).

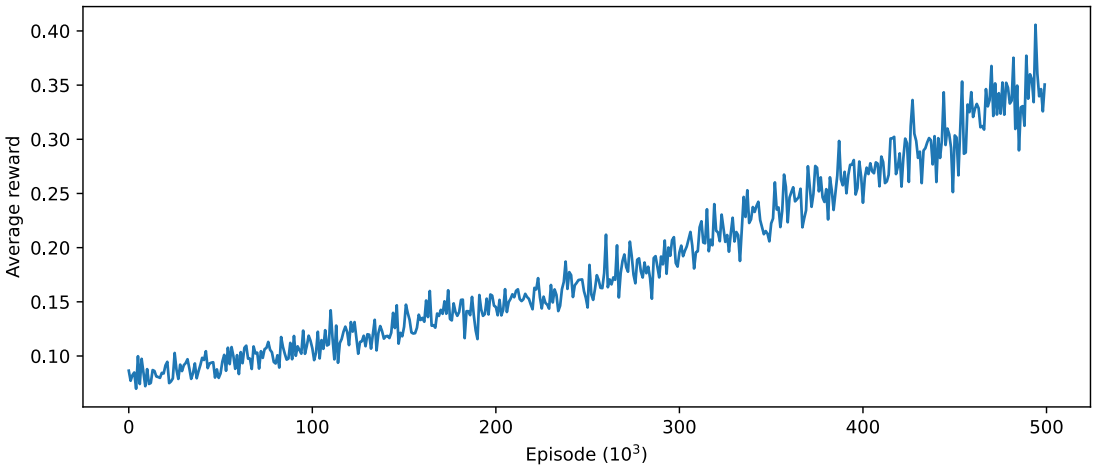
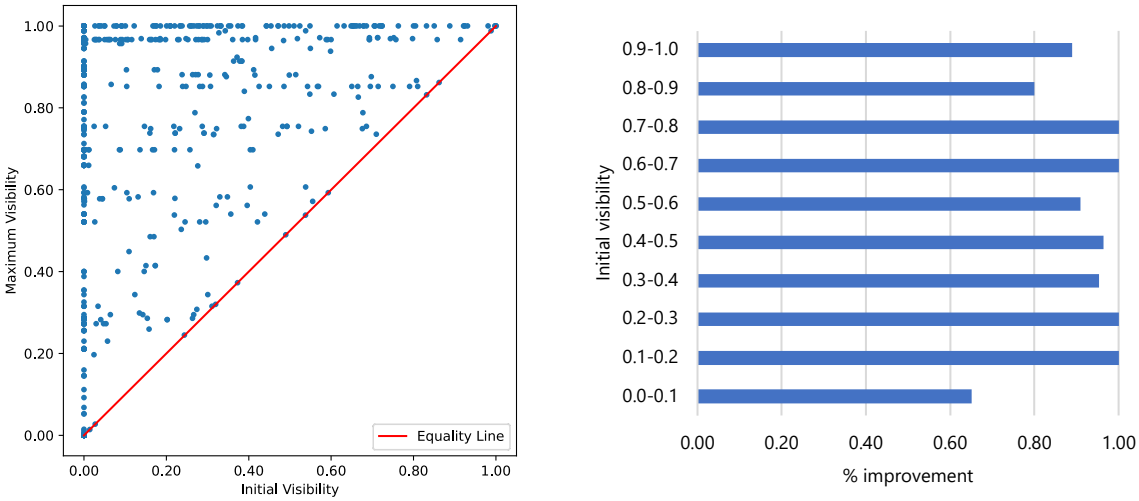


Figure 3: Average reward received in 1,000-episode intervals during training.



(a) (b)

Figure 4: Improvement in visibility for finding forklifts in the warehouse, for various initial positions of the camera – (a) initial vs. maximum visibility, and (b) % improvement for various initial visibilities.

5. Summary and Conclusion

Continuous and unobtrusive monitoring of jobsite activities is critical for safety monitoring, supply chain management, asset management, resource allocation, and productivity assessment. In this paper, we introduced a self-navigating camera capable of finding a moving object of interest (i.e., forklift) by autonomously browsing a real-world warehouse environment. We

trained a DQN with RL to provide situational awareness to the camera. Particularly, the RL agent was trained to take one of the five actions of moving "up", "down", "left", "right", or "do nothing" to navigate through the environment. The value of reward was determined based on the IoF between the bounding box of the forklift and the viewing window of the RL camera. The model was validated with a 360-degree video of a real-world warehouse operation. Results indicated that starting from a random initial position, the RL agent was able to successfully find a moving forklift by autonomously browsing the warehouse environment, and in 73% of the times, improve the visibility of the tracked object. This, for example, exceeds the 71% true positive rate in detecting objects in still images, as reported by Caicedo and Lazebnik (2015).

One of the major challenges in conducting RL experiments in real-world settings is to evaluate the models thoroughly and safely without compromising the safety of human participants or causing unwanted damage to physical assets. Therefore, a potential direction of future work will be to conduct experiments in controlled real-world environments as a first step toward improving the performance of RL models in a wide variety of applications such as construction safety, warehouse operations, smart spatiotemporal surveillance, and post-disaster search and rescue operations. Another potential implementation challenge is the inference time which can undermine the usefulness of the developed RL models for real-time decision-making. A potential way to reduce inference time is to perform neural network pruning by reducing redundant layers (Pi et al., 2021).

References

Autodesk (2020). Future of Work, Autodesk website, https://www.autodesk.com/future-of-work, accessed April 2022.

Busoniu, L., Babuska, R., De Schutter, B., Ernst, D. (2010). Reinforcement learning and dynamic programming using function approximators. CRC press.

Caicedo, J. C., Lazebnik, S. (2015). Active object localization with deep reinforcement learning. In: IEEE International Conference on Computer Vision (ICCV), 2015, Santiago, Chile.

Canning, J. (2020). 1.43 trillion photos were taken in 2020 but how many of them were captured on our mobile phones?, Buymobile website, https://www.buymobiles.net/blog/1-43-trillion-photos-were-taken-in-2020-but-how-many-of-them-were-captured-on-our-mobile-phones, accessed April 2022.

Chollet, F. (2021). Deep Learning with Python. Simon and Schuster.

Deng, L., Liu, Y. (2018). Deep learning in natural language processing. Springer.

Dos Santos, S. R. B., Givigi, S. N., Nascimento, C. L. (2013). Autonomous construction of structures in a dynamic environment using reinforcement learning, In: IEEE International Systems Conference (SysCon), 2013, Orlando, FL.

Ekenel, H. K., Stiefelhagen, R. (2009). Why is facial occlusion a challenging problem?, In: Third International Conference on Advances in Biometrics, 2009, Alghero, Italy.

Fang, Q., Li, H., Luo, X., Ding, L., Luo, H., Rose, T. M., An, W. (2018). Detecting non-hardhat-use by a deep learning method from far-field surveillance videos, J. Automation in Construction. 85, pp. 1–9.

Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. Sebastopol, CA, O'Reilly Media.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D. (2018). Deep reinforcement learning that matters, In: AAAI Conference on Artificial Intelligence, 2018, New Orleans, LA.

Hoiem, D., Efros, A. A., Hebert, M. (2011). Recovering occlusion boundaries from an image, International J. Computer Vision. 91(3), pp. 328–346.

Jang, B., Kim, M., Harerimana, G., Kim, J. W. (2019). Q-learning algorithms: A comprehensive classification and applications. IEEE Access 7:133653-133667.

Krizhevsky, A., Sutskever, I., Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM. 60(6), pp. 84–90

Kormushev, P., Calinon, S., Caldwell, D. G. (2013). Reinforcement learning in robotics: Applications and real-world challenges, J. Robotics. 2(3), pp. 122–148.

Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.

Lin, Y., Dai, X., Li, L., Wang, F.-Y. (2018). An efficient deep reinforcement learning model for urban traffic control. arXiv preprint arXiv:1808.01876.

Mccoy, L. G., Nagaraj, S., Morgado, F., Harish, V., Das, S., Celi, L. A. (2020). What do medical students actually need to know about artificial intelligence?, J. Digital Medicine. 3(1), pp. 1–3.

Mullapudi, A., Lewis, M. J., Gruden, C. L., Kerkez, B. (2020). Deep reinforcement learning for the real time control of stormwater systems, J. Advances in Water Resources. 140, pp. 103600.

Muresan, M., Fu, L., Pan, G. (2019). Adaptive traffic signal control with deep reinforcement learning an exploratory investigation. arXiv preprint arXiv:1901.00960.

Nath, N. D., Behzadan, A. H. (2020a). Deep convolutional networks for construction object detection under different visual conditions, J. Frontiers in Built Environment. 6(97), pp. 1–22.

Nath, N. D., Behzadan, A. H. (2020b). Deep generative adversarial network to enhance image quality for fast object detection in construction sites. In: IEEE Winter Simulation Conference (WSC), 2020, Orlando, FL.

Nath, N. D., Behzadan, A. H., Paal, S. G. (2020). Deep learning for site safety: Real-time detection of personal protective equipment, J. Automation in Construction. 112, pp. 103085.

Nguyen, T. T., Nguyen, N. D., Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications, IEEE Transactions on Cybernetics. 50(9), pp. 3826–3839.

Nian, R., Liu, J., Huang, B. (2020). A review on reinforcement learning: Introduction and applications in industrial process control, Computers and Chemical Engineering. 139, pp. 106886.

NSF (2020). Future of Work at the Human-Technology Frontier, United States National Science Foundation website, https://www.nsf.gov/news/special_reports/big_ideas/human_tech.jsp, accessed April 2022.

OSHA (2019). Commonly Used Statistics, Unite States Department of Labor website, https://www.osha.gov/oshstats/commonstats.html, accessed April 2022.

Pi, Y., Nath, N. D., Sampathkumar, S., Behzadan, A. H. (2021). Deep learning for visual analytics of the spread of COVID-19 infection in crowded urban environments, J. Natural Hazards Review. 22(3), pp. 04021019.

Rao, Q., Frtunikj, J. (2018). Deep learning for self-driving cars: Chances and challenges. In: 1st International Workshop on Software Engineering for AI in Autonomous Systems, 2018, Gothenburg, Sweden.

Rasheed, F., Yau, K.-L. A., Low, Y.-C. (2020). Deep reinforcement learning for traffic signal control under disturbances: A case study on Sunway City, Malaysia, J. Future Generation Computer Systems. 109, pp. 431–445.

Schaul, T., Quan, J., Antonoglou, I., Silver, D. (2015). Prioritized experience replay. arXiv preprint arXiv:1511.05952.

Shan, Q., Li, Z., Jia, J., Tang, C.-K. (2008). Fast image/video upsampling, ACM Transactions on Graphics (TOG). 27(5), pp. 1–7.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. J. N. (2016). Mastering the game of Go with deep neural networks and tree search, J. Nature. 529(7587), pp. 484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. J. N. (2017). Mastering the game of go without human knowledge, J. Nature. 550(7676), pp. 354 –359.

Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Sowa, K., Przegalinska, A., Ciechanowski, L. (2021). Cobots in knowledge work: Human-AI collaboration in managerial professions, J. Business Research. 125, pp. 135–142.

Sutton, R. S., Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Szepesvári, C. (2010). Algorithms for reinforcement learning. San Rafael, CA, Morgan & Claypool Publishers.

Szita, I. (2012). Reinforcement learning in games. Reinforcement learning. Springer, pp. 539–577.

Tan, K. L., Poddar, S., Sarkar, S., Sharma, A. (2019). Deep reinforcement learning for adaptive traffic signal control. In: ASME Dynamic Systems and Control Conference. 59162, V003T18A006.

Tschandl, P., Rinner, C., Apalla, Z., Argenziano, G., Codella, N., Halpern, A., Janda, M., Lallas, A., Longo, C., Malvehy, J. (2020). Human-computer collaboration for skin cancer recognition, J. Nature Medicine. 26(8), pp. 1229–1234.

Wang, D., Weisz, J. D., Muller, M., Ram, P., Geyer, W., Dugan, C., Tausczik, Y., Samulowitz, H., Gray, A. (2019). Human-AI collaboration in data science: Exploring data scientists' perceptions of automated AI. Proceedings of the ACM on Human-Computer Interaction, 3, pp. 1–24.

Wu, Y., Tan, H., Ran, B. (2018). Differential variable speed limits control for freeway recurrent bottlenecks via deep reinforcement learning, arXiv preprint arXiv:1810.10952.

Xu, M., Wu, J., Huang, L., Zhou, R., Wang, T., Hu, D. (2020). Network-wide traffic signal control based on the discovery of critical nodes and deep reinforcement learning, J. Intelligent Transportation Systems. 24(1), pp. 1–10.

Yao, L., Dong, Q., Jiang, J., Ni, F. (2020). Deep reinforcement learning for long-term pavement maintenance planning, J. Computer-Aided Civil Infrastructure Engineering. 35(11), pp. 1230–1245.

Zhang, D., Han, X., Deng, C. (2018). Review on the research and practice of deep learning and reinforcement learning in smart grids, CSEE J. Power Energy Systems. 4(3), pp. 362–370.