TEACHING ROBOTICS DURING COVID-19: MACHINE LEARNING, SIMULATION, AND AWS DEEPRACER

Peter Holowka
West Point Grey Academy, Vancouver, British Columbia, Canada

ABSTRACT

COVID-19 presented a challenge to the traditional methods of teaching programming and robotics in a secondary school environment. When campuses were closed around the world in the spring of 2020, it was not possible for students to access the computer labs nor the robotics equipment that was traditionally used to facilitate the instruction of robotics programming units. This paper presents a design research project in which two secondary institutions in Canada and Turkey collaborated to teach computer science and robotics programming, specifically reinforcement learning, through the use of an online simulation environment. The two student cohorts in the study both were successful in developing reinforcement learning models for autonomous vehicles, despite not having any prior experience in machine learning nor artificial intelligence. The implications of this work are that physical robotics kits and dedicated robotics spaces are not essential to the teaching of programming and robotics. This is especially relevant to marginalized communities that do not have the resources to support robotics instruction, further exacerbating the digital divide.

KEYWORDS

Machine Learning, Artificial Intelligence, AWS DeepRacer, COVID-19, Simulation, Education

1. INTRODUCTION

The concept of Just in Time Learning (JITL) was initially primarily found in the context of professional and/or technical knowledge (Boese, 2016). The academic literature on this topic has recently expanded to include more formal education, including primary, secondary, and tertiary education (Khamis et al., 2019). JITL is defined as the acquisition of knowledge in response to an imminent problem or context requiring that knowledge. This paper will discuss the application of JITL as a means of educating and motivating secondary school students to learn Artificial Intelligence (AI) and Machine Learning (ML) during COVID-19. This work was conducted in the spring of 2020, during the height of the COVID-19 outbreak, within the challenging context of emergency remote instruction when access to physical robotics resources and computer labs was prohibited. Novice computer science students were successful in programming autonomous robotics vehicles, as well as learning the fundamentals of the Python programming language through the JITL approach. Moreover, the simulation environment enabled students to explore advanced topics in AI and ML as the simulation environment was free of many of the challenges that are characteristic of teaching with physical robotics kits. The advantages of using a simulation environment over physical robotics kits in a computer lab will be discussed further later in this paper.

1.1 Building upon a Foundation of Cloud Computing

In the spring of 2020, in response to the COVID-19 pandemic, physical school campuses were closed and students were required to attend school remotely. For many students, attending school meant interacting teachers through an online environment over the internet. This was the experience of the two cohorts in this design-based research project (Design-Based Research Collective, 2003; Zydney et al., 2020). One cohort was from a secondary school in Vancouver, Canada and consisted of seven students in the ninth grade. The other cohort was from a secondary school in Istanbul, Turkey and consisted of five students in the tenth and eleventh grades. The robotics and computer programming instruction for these two cohorts was delivered exclusively online using cloud computing. All students worked from their homes without access to school

computers, robotics equipment, or specialized/licensed software. A combination of cloud computing tools was used to facilitate instruction. Zoom was used for synchronous video communication, hosted by the teacher. Google Classroom, Google Drive, and Google Docs were used for asynchronous document sharing and collaboration. The AI and ML simulation environment, AWS DeepRacer, was hosted on Amazon's public cloud platform as well.

This paper extends upon the author's earlier work concerning cloud computing in education within a Western Canadian context. The foundational work was an exhaustive study of all 75 large K-12 districts in Canada's three western-most provinces (Holowka , 2018). The study employed a data transformation mixed-methods triangulation design and captured the educational IT infrastructure serving over 1.14 million students (Creswell et al., 2003). The findings of the study revealed that Western Canada was a leader in the adoption of cloud computing globally with every school district employing several forms of cloud computing for their IT infrastructure. This early adoption of cloud computing now enables future cloud computing technologies, such as the online simulation environment for teaching AI and ML discussed in this paper. Moreover, the ubiquity of cloud computing in Western Canadian K-12 underscores the broad adoption possibilities for this simulation technology due to the existing IT infrastructure already available to students and their institutions. Similarly, the consistent internet access of the Istanbul cohort, as well as their success with this project, suggest that a shift to cloud computing-enabled instructional tools is an option for many schools around the world.

1.2 AWS DeepRacer

Amazon Web Services (AWS) AWS DeepRacer consists of two components: a physical 1/18th scale race car and an online simulation environment for programming the self-driving performance of the small scale race car (Balaji et al., 2019). Using AI and ML, users of the AWS DeepRacer console customize self-driving reinforcement learning models which they develop in the online simulation environment. These models can be transferred to the physical car for real-world testing, though this real-world testing is not required. Figure 1 shows the physical 1/18th scale race cars that contain a Linux operating system and a computer vision camera, in addition to a typical toy car electric motor drivetrain. Within the context of this paper, the use of AWS DeepRacer will refer to programming within the online simulation environment. This work did not use the physical race cars as they were inaccessible at the school due to COVID-19 restrictions. A discussion of the use of the physical cars occurs in the concluding section on future opportunities for research.

The AWS DeepRacer service incorporates multiple AWS services in order to create an environment for the development and testing of autonomous vehicles. These multiple AWS services include storage (AWS S3), compute (EC2), virtual environment creation tools (AWS Robomaker), and so forth (Balaji et al., 2019; Suenaga & Morioka, 2020). The integration of these discrete services into the single AWS DeepRacer service helps students learn complex AI and ML concepts with greater ease. If learning to program AI and ML can be likend to learning to drive a real car, then the use of AWS DeepRacer allows students to concentrate on their use of the brake pedal, rather than first trying to have them understand the theory behind how disc brakes work. The use of AWS DeepRacer in a secondary school educational setting embodies the JITL approach. Using AWS DeepRacer and JITL, students can focus on an immediate outcome they are trying to achieve, rather than being overwhelmed by the wide-ranging supporting individual components.

The use of AWS DeepRacer and JITL can be viewed as an alternative or modification of the conventional scaffolding approach used by educators. The classic use of scaffolding consists of students being guided through increasingly complex concepts and skills, with earlier work serving as a scaffolding for latter work (Hogan & Pressley, 1997). In this AWS DeepRacer application of JITL, students began with the seemingly complex task of programming a self-driving car. At the beginning, students had only a rudimentary understanding of how the multiple underlying AWS technologies functioned. However, with experience using the AWS DeepRacer environment, students gained familiarity with the integrated systems of the AWS DeepRacer plaform. Students are then encouraged to further optimize their cars' performance by more deeply investigating the individual supporting components of the integrated system. Similar to reverse engineering, students were invited to experiment by adjusting the underlying components. This AWS DeepRacer project revealed that once students gained an understanding of the basic operation of their model and the simulation environment, they were then motivated to investigate how the modification of individual supporting components could enhance their work.

Figure 1. Physical AWS DeepRacer cars that can run ML models trained in an online simulator

2. PROJECT GOALS AND STUDENT SKILLS

2.1 Cloud Computing Infrastructure

The purpose of the AWS DeepRacer unit was to introduce students to the fundamentals of computer science. This included an overview of cloud computing infrastructure and programming languages. The project began with the creation of individual student accounts, by the students themselves, in the AWS Console. Students created full AWS accounts and were introduced to the login process and the various services available by the instructor. Students were also given a lecture on ethics and the importance of cost management when using cloud computing services. The configuration of AWS DeepRacer was an authentic lesson about cloud computing using cloud computing.

2.2 Programming Languages and Python

The underlying motivation for the course and the AWS DeepRacer unit was to introduce students without prior programming experience to the fundamentals of computer science, namely to develop an understanding of programming languages such as Python. AWS DeepRacer provides a low-code environment where students can achieve success by using pre-populated reward function code. Students were first encouraged to pursue success by any approach, including the use of the pre-populated code library. This would serve as a baseline for student learning. Students were able to successfully complete the course task without fully understanding programming languages. This was the baseline and the aim for students was the mere completion of a simple oval track within the AWS DeepRacer simulator. Figure 2 presents the simulation environment for a simple oval track. Figure 3, in contrast, presents the simulation environment for a more complex track that students later used.

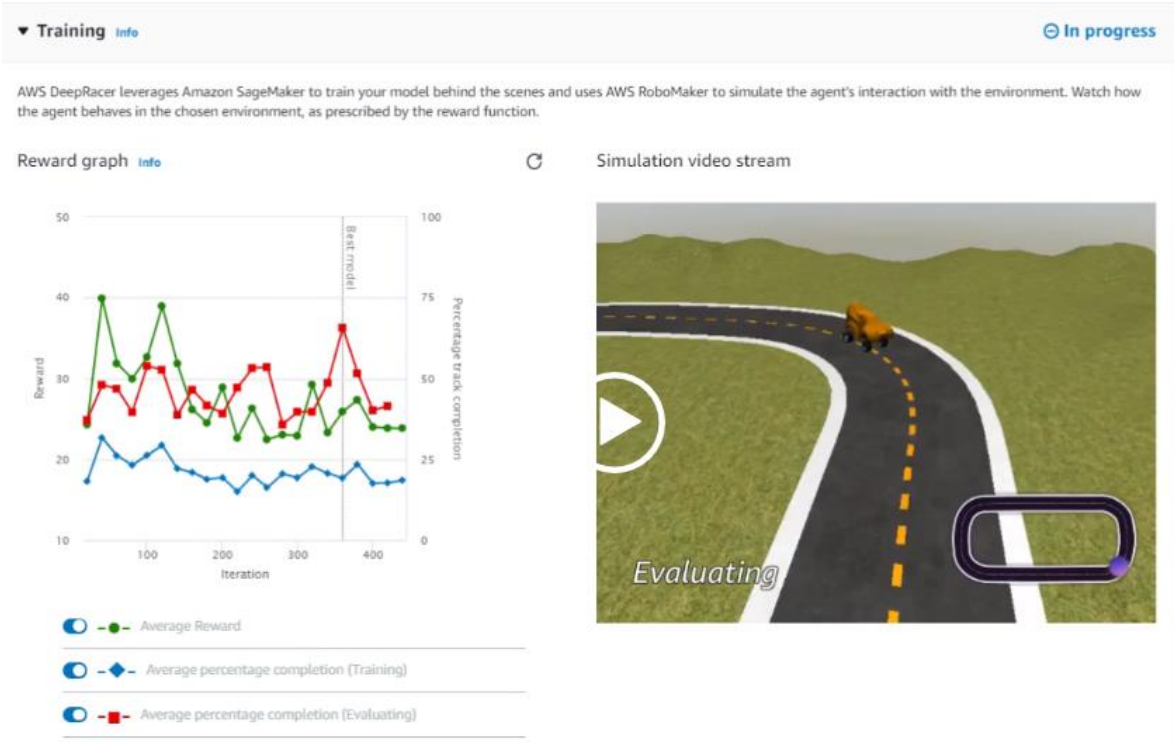

Figure 2. AWS DeepRacer model training and evaluation within simulation on a simple oval track

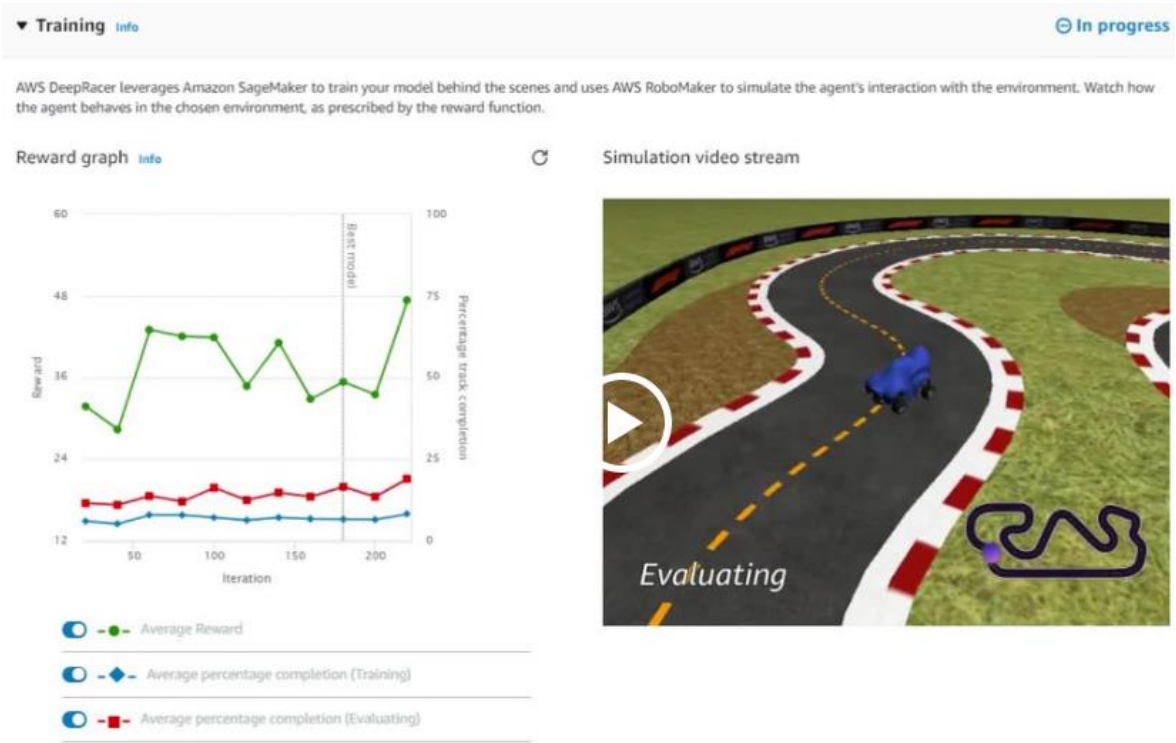

Figure 3. AWS DeepRacer model training and evaluation within simulation on a complex track

Using the synchronous and asynchronous cloud computing collaboration tools outlined earlier, students in the Vancouver and Istanbul cohorts exchanged ideas concerning how to improve their models and how to more quickly complete the simple oval track. Students reviewed the sample reward function code provided in the Python programming language and began to understand the code's structure. Similarly, they identified variable that they could change. An example of the Python reward function code for a successful reinforcement learning model is found in Figure 4. This code was initially developed for the simple oval track in Figure 2 but was then transferred to the more complex track in Figure 3.

Reward function

```python
def reward_function(params):
    '''
    Example of rewarding the agent to follow center line
    '''

    # Read input parameters
    all_wheels_on_track = params['all_wheels_on_track']
    track_width = params['track_width']
    distance_from_center = params['distance_from_center']
    steps = params['steps']
    progress = params['progress']

    # Calculate 3 markers that are at varying distances away from the center line
    marker_1 = 0.01 * track_width
    marker_2 = 0.08 * track_width
    marker_3 = 0.2 * track_width
    marker_4 = 0.3 * track_width
    marker_5 = 0.5 * track_width

    #if all_wheels_on_track and (0.5*track_width - distance_from_center) >= 0.05:
    #    reward = 1.0

    # Total num of steps we want the car to finish the lap, it will vary depends on the track length
    TOTAL_NUM_STEPS = 10

    if (steps % 100) == 0 and progress > (steps / TOTAL_NUM_STEPS) * 100 :
        reward += 10.0

    # Give higher reward if the car is closer to center line and vice versa
    if all_wheels_on_track and distance_from_center <= marker_1:
        reward = 1.0
    elif all_wheels_on_track and distance_from_center <= marker_2:
        reward = 0.5
    elif distance_from_center <= marker_3:
        reward = 0.2
    elif distance_from_center <= marker_4:
        reward = 0.1
    elif distance_from_center <= marker_5:
        reward = 0.001
    else:
        reward = 1e-3  # likely crashed/ close to off track

    return float(reward)
```

Figure 4. An example of a successful AWS DeepRacer model reward function written in Python

2.3 Student Collaboration, Competition, and Success

The underlying goal of the AWS DeepRacer unit was to teach students robotics and programming skills. This was successful as each student in the course was able to iterate upon their reinforcement learning models and demonstrate an increased understanding of the Python programming language. By the end of the AWS DeepRacer unit, students were required to complete the following three tasks: 1) to successfully complete five virtual races on five different tracks within the simulation environment, 2) to develop a log of changes to

their reward functions written in Python, and 3) provide an optional video reflection on their experience collaborating with peers, both locally and internationally. All students successfully completed the first two tasks and the majority of students completed the third.

The students demonstrated considerable growth in their understanding of the Python programming language and pursued advanced topics beyond the course's scope in order to achieve a higher position on the races' leaderboards. When students completed a track, their lap times were recorded automatically within the AWS DeepRacer Community Race Leaderboard. The competitive element of the activity, even though it did not have any relation to the course mark, provided added motivation and engagement for the students. Students took pride in leapfrogging each others' performance as they learned more advanced reinforcement learning strategies. Moreover, the leaderboards indicated the completion lap times for the students and was a helpful tool for showing the improvements in the performance of their AI/ML models.

An enabler of students' success was the open, collaborative nature of the activity. Students were required to produce a shared development log which detailed both successful ML models, as well as models which resulted in failure. The instructor modeled this activity by providing an open example for students to view using a shared Google Doc. The instructor included both working reward functions and reward functions that were unsuccessful. This openness modeled the collaborative scholarly process of research, as well as provided students with time-saving insights as to what strategies to avoid. The collaborative and social nature of this activity was often described as a positive aspect of the AWS DeepRacer exercise by the students in their reflection videos.

3. REIMAGINING ROBOTICS INSTRUCTION

COVID-19 exacerbated existing challenges pertaining to teaching robotics within a secondary school context. Prior to COVID-19, teaching computer programming for robotics was problematic because of the physical space required for the use of traditional robotics kits. When schools were moved to fully online instruction in the spring of 2020, access to such kits was further restricted. Prior to COVID-19, the Vancouver school shifted away from traditional robotics kits towards a cloud-based simulation environment for teaching programming and robotics. This approach resolved many of the challenges associated with teaching programming and robotics and made possible the international collaboration described in this paper. The success of this collaboration now serves as a guide, informing the teaching of programming and robotics in upcoming academic years.

3.1 Proprietary Physical Robotics Kit Limitations

The conventional means of teaching computer programming and robotics in many primary and secondary schools involves the use of physical robotics kits. These kits often contain proprietary components and kit/vendor-specific software that is not widely used in the IT industry (Vandevelde et al., 2013). Examples of such proprietary robotics kits include those made by VEX Robotics and Lego Mindstorms (Habib, 2012). Such kits are problematic for some schools in that they require a considerable initial financial outlay, physical storage space within the school, computers with licensed software through which to update the proprietary robotics controller hardware, maintenance of the computers which are used for the licensed software, maintenance/replacement of the robotics kits' parts, and consistent access to physical space for students to experiment with the physical robots. In many schools, the availability of physical space for such robotics activities would be at the expense of space for the enrollment of students. The challenges associated with physical robotics kits are considerable and can present a barrier to schools having any kind of robotics program and/or programming club.

3.2 Benefits of an Online Simulation Environment for Robotics Instruction

An online simulation environment for robotics instruction is free from all of the challenges associated with physical robotics kits outlined in the previous section. The shift from physical robotics kits towards virtual robots within an online simulation hosted in the cloud also yields benefits pertaining to instructional design. In addition to the space and equipment required for physical robotics kits, a considerable percentage of an

activity's time involving such kits is often dedicated to the management of the physical components of the robot (e.g., plastic gears, wheels, etc.). The students' efforts go towards assembling and repairing the pieces of the robot, rather than developing and refining the code. In such an example, a considerable amount of time is spent by students on low-level assembly work rather than on the intended higher-level analysis of their code. The use of an online simulation environment means that students work nearly entirely on the programming of their vehicles and the code of their reward functions.

The use of an online simulation environment using cloud computing also allows students to work on their robotics projects at any time from any location with internet access. Students are able to log into the simulation environment from a web browser, such as Google Chrome, without the need for any other specialized software. Students can then experiment with the code of their model and run virtual training/evaluation sessions where they test the code. Educational initiatives by vendors, such as AWS Educate, give students free computational credits/hours resulting in no additional cost to students or their institutions.

As the testing of the code can take hours, students can also run these training/evaluation sessions unattended. Students can create a model training/evaluation, turn off their computer, and then review the results hours later. This is possible because the simulation runs entirely in the cloud, independent of the students' computer. This allows students to advance their robotics work at home and at school. This was a critical benefit during COVID-19 when schools were inaccessible. The simulation environment allowed students to engage fully in the robotics unit, in spite of not being able to access the school buildings.

3.3 Online Simulations Leading to Physical Robotics Testing

The use of online simulations for teaching students programming and robotics is meant to expedite the students' learning and not to serve as a complete replacement for physical robotics programming. The reality gap in robotics refers to the differences between simulations and reality (Collins et al., 2019; Koos et al., 2012). The AWS DeepRacer simulation environment is a training environment for reinforcement learning models which can later be downloaded to a robust physical small race car (Figure 1). Though this project did not involve the use of these physical cars, it is important to clarify that the online simulation environment is an accelerant for the development of student skills with robotics, rather than a purely virtual replacement with no connection to the real world. When students are ready, they can transfer their reinforcement learning models to the physical cars. The ability to explore and address the reality gap will be a valuable opportunity to extend their learning in the fields of robotics and programming.

4. CONCLUSION

The use of AWS DeepRacer during COVID-19 within a secondary school context was an emergency initiative that revealed the viability of using a simulation environment to teach a non-expert audience robotics, programming, AI, and ML. The initiative was successful in that all students were able to program working models for their simulation environments and complete multiple races/tracks. Students who had no prior experience with the Python language were able to modify/create the reinforcement learning code to improve their AI/ML car models. Students also engaged with concepts outside of the course's core curriculum in efforts to improve their models. This included modifying AI/ML hyperparameters and researching senior-level calculus to optimize the reward functions they had written in Python. As the majority of the students were in the ninth grade, this early effort to understand senior-level concepts in mathematics will hopefully allow them to make connections to the more advanced/abstract mathematical concepts they will encounter in future courses.

Limitations of this work pertain primarily to this project's relatively small size. As this was a pilot project, conducted in an emergency setting during the COVID-19, there are opportunities to expand upon this work in the future. Future studies can focus on replicating and expanding this study to better understand the efficacy of robotics instruction within a simulation environment. For example, a more experimental research design can be implemented where two cohorts of students are taught similar concepts of AI and ML. One cohort could be taught using a simulation environment such as AWS DeepRacer, while the other could be taught using more conventional, on-site-only methods with physical robotics kits. Quantitative analysis could

be applied to examine the differences between the two groups, especially if multiple large cohorts can be engaged. Similarly, future research can be conducted into the reality gap and the challenges associated with transferring self-driving models from the AWS DeepRacer simulation environment to the physical cars.

A further opportunity for a future study may involve a simple replication of this work with a different cohort of students. It is possible that the success of this project was enabled by the very positive and perhaps exceptional nature of the students in this initial study's cohort. Future studies with different students, schools, and teachers would also yield valuable insights into the efficacy of using simulation environments to teach foundational robotics programming concepts.

ACKNOWLEDGEMENT

The author would like express his sincere gratitude to the two schools that participated in this project: West Point Grey Academy in Vancouver, British Columbia and Bahçeşehir Koleji in Istanbul, Turkey.

REFERENCES

Balaji, B., Mallya, S., Genc, S., Gupta, S., Dirac, L., Khare, V., Roy, G., Sun, T., Tao, Y., Townsend, B. and Calleja, E., 2019. AWS DeepRacer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning. *arXiv preprint arXiv:1911.01562*.

Boese, E., 2016, February. Just-In-Time learning for the just Google it era. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 341-345).

Collins, J., Howard, D. and Leitner, J., 2019, May. Quantifying the reality gap in robotic manipulation tasks. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 6706-6712). IEEE.

Creswell, J.W., Plano Clark, V.L., Gutmann, M.L. and Hanson, W.E., 2003. Advanced mixed methods research designs. *Handbook of mixed methods in social and behavioral research*, *209*(240), pp.209-240.

Design-Based Research Collective, 2003. Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, *32*(1), pp.5-8.

Habib, M.A., 2012. Starting a robotics program in your county. *Journal of Extension*, *50*(2), p.20.

Hogan, K.E. and Pressley, M.E., 1997. *Scaffolding student learning: Instructional approaches and issues*. Brookline Books. Cambridge, MA.

Holowka, P., 2018. 'IT leadership and cloud computing adoption in western Canadian K-12 school districts', doctoral dissertation, University of Calgary, Calgary, AB, DOI:10.11575/PRISM/32649.

Khamis, T., Gyn, G.V. and Rarieya, J., 2019. Introduction to the special issue of SOTL in the South: advancing student engagement in learning–experiences from Pakistan.

Koos, S., Mouret, J.B. and Doncieux, S., 2012. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, *17*(1), pp.122-145.

Suenaga, R. and Morioka, K., 2020, January. Development of a Web-Based Education System for Deep Reinforcement Learning-Based Autonomous Mobile Robot Navigation in Real World. In *2020 IEEE/SICE International Symposium on System Integration (SII)* (pp. 1040-1045). IEEE.

Vandevelde, C., Saldien, J., Ciocci, M.C. and Vanderborght, B., 2013. Overview of technologies for building robots in the classroom. In *International Conference on Robotics in Education* (pp. 122-130).

Zydney, J.M., Warner, Z. and Angelone, L., 2020. Learning through experience: Using design based research to redesign protocols for blended synchronous learning environments. *Computers & Education*, *143*, p.103678.