Applications of Q-Learning in Coding with Snap!

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Adam Emerson Marcus

Spring 2021

Technical Project Team Members

Ethan Gumabay

Eric Stein

Yuxin Wu

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Rich Nguyen, Department of Computer Science

In today's society, technology is a part of everyday life for most people. As a result, it is a widespread belief that "different sectors should be represented in the process to define and develop technology" (Garcia-Holgado et al. 2019). While it would be ideal if there was interest in pursuing careers in technology from every person, that is simply not the case. According to a study by Tai, Liu, Maltese, and Fan, "students with expectations for a science related career were 3.4 times more likely to earn physical science and engineering degrees that students without similar expectations" (2006). After seeing this information, the question becomes one of how to engage students that are pursuing other fields so that people similar to them might be represented in the design and development of future technologies. There are currently some technologies with this in mind such as Arduino, a company that produces microcontrollers and microcontroller kits that enable users to do many different things such as create wearable technologies by sewing the microcontroller and sensors into fabrics (Beuchley et al., 2008). This concept can be expanded into other fields in the arts besides clothing design, such as drawing or creating music.

During the 2020-2021 academic year, I will be working on a capstone project in a team with two undergraduate students, Eric Stein and Yuxin Wu, as well as one graduate student, Ethan Gumabay. Our technical advisor will be Rich Nguyen, a machine learning professor in the Department of Computer Science. In addition, Glen Bull, a professor in the Curry School of Education will be working with us at guiding the overall goals of our project. For our project, we are working on several different facets of a new educational tool for professors to teach computer science. It is built on an open source platform, Snap!, which was created at the University of California Berkley as a tool to simplify and teach the basics of coding to new computer science students (Harvey & Mönig, 2020). The project we're working on is called

Tunescope, and it aims to provide students a way to integrate computer science education with the arts. Using Snap! we have created our own custom blocks for students to use that allow them to code both music and art in an attempt to spice up computer science education.

Figure 1. Example Blocks Used by Students in Snap! (Harvey & Mönig, 2020)

Tunescope is currently used in a seminar taught by Glen Bull that focuses on learning how students can use coding to create music and art as two of the pieces of the course. This seminar also uses the Arduino devices mentioned before. My individual goal for this project focuses on a more specific feature: how students receive help and feedback for specific tasks. I completed a proof of concept for the applications of machine learning in our project for a simple assignment. The task my project focuses on consists of having the students draw a square using the Snap! framework provided. To achieve this goal, I created a Q Learning Agent that takes in the current set of code blocks selected by a student and is able to suggest the best action for that student to take in order to complete their task of drawing a square. This functionality in our project for such a simple problem is designed to show that machine learning can be used to help not just students who are stuck but also professors who don't have the time and/or knowledge to give each student personalized feedback at any point in their work.

After creating and training the Q Learning Agent, incorporating such a piece into the Tunescope website to help students presented some challenges. A Q Learning Agent relies on a

Q-table to predict the best action to take in any given state. Given the open ended nature of coding with Snap!, the state space (set of all potential code block permutations) and action space (set of all potential changes you could make to a set of code blocks) are infinite and therefore must be restricted for the specific problem of drawing a square. Even with these restrictions, the number of states and actions is enormous. In order to get hints from the Q Learning Agent, one must simply look up the best action in its Q-table. This prompts the largest obstacle in widespread use of Q learning: Q-tables for such problems can get too large very quickly, making them inefficient in terms of overhead. Simply uploading the data in the trained Q-table to a place accessible by the Tunescope website took several hours to complete. Due to this inefficiency, after completing the Q Learning Agent, I shifted to approach this problem in a better way.

Rather than using the simpler Q Learning Agent, I employed a deep Q-network (DQN) as a means of predicting the best action to take in any state, without the absurd storage requirements of saving and updating a Q-table. The main difference between the Q learning approach and the DQN approach is that the agent chooses actions based on a Q-table with distinct values versus using approximations from training a Q-network. Using TensorFlow's premade DQN agent class, I tested several different Q-networks using various learning rates, dense layers, and activation functions, comparing the loss and returns for each until I identified the best hyperparameters for the agent to use. Currently this piece of my project is still in the development phase as I work to tune the hyperparameters to further improve accuracy.

The Tunescope website has a custom code block that students can click, called "Get Hint" which takes in their current set of code blocks and returns a hint from the Q-table providing them the best action they could take towards having their code successfully draw a square. Once the DQN agent has been improved further, the hints will instead be retrieved from

the DQN agent's predictions, instead of simply looking them up in the Q-table. The next steps beyond the implementation of a DQN agent for solving the problem of drawing a square include expanding the issues tackled to other problems that students will be shown in the course taught by Professor Bull. The versatility of a DQN agent can be shown off in such situations, as the low overhead will allow us to train separate agents for any similar problem while only making minor changes regarding what the goal state of students' code looks like for each one.

References

Buechley, L., Eisenberg, M., Catchen, J., & Crockett, A. (2008, April). The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 423-432).

Garcia-Holgado, A., Vázquez-Ingelmo, A., Verdugo-Castro, S., González, C., Gómez, M. C. S., & Garcia-Peñalvo, F. J. (2019, April). Actions to promote diversity in engineering studies: a case study in a Computer Science Degree. In *2019 IEEE Global Engineering Education Conference (EDUCON)* (pp. 793-800). IEEE.

Harvey, B., & Mönig, J. (2020). Snap! Reference Manual. Retrieved November 04, 2020, from https://snap.berkeley.edu/snap/help/SnapManual.pdf

Tai, R. H., Liu, C. Q., Maltese, A. V., & Fan, X. (2006). Planning early for careers in science. *Life sci*, *1*(0.2).