Introduction and Game Mechanics

StarCraft II is the sequel to Starcraft, a timeless real-time strategy game that takes place in a science-fiction universe ruled by three balanced races: Terran, Zerg, and Protoss (abbreviated as T, Z, and P). For the purpose of this project, we will limit our focus to strictly StarCraft II, specifically, competitive 1v1 games. The goal of each game is simple: efficiently allocate resources into building an army in order to defeat the opponent’s base.

At the start of each game, both players start out with one base and twelve workers, units that can mine resources and create buildings. There are two resources in the game: minerals and vespene gas. Once harvested, these resources can be used to build army units, buildings, and additional workers. A player can boost their resource collection rate in the long run by building more bases and workers during the early stages of the game. However, this forgoes the opportunity to build offensive units, which leaves a player vulnerable to attacks from the enemy. The challenge, then, for each player is to balance collecting resources and building a sufficiently-sized army.

The keyboard and the mouse is the portal between players’ minds and their respective units. Since the game is played in real time, speed and effective multitasking abilities are imperative. This is especially true late in the game, when players must control hundreds of units and dozens of buildings. Professional StarCraft II players average up to 300 to 400 actions per minute.

StarCraft II also includes a game mechanic known as “fog of war”, which limits the vision of each player to a small radius around each of their units and buildings. As a result, each player has limited information about their opponent’s strategy, including their allocation of resources, location of army, and buildings created. It is up to the player to use the vision of his or her units to deduce the opponent’s strategy. The fog of war mechanic is one factor that makes Starcraft II such a complicated game for artificial intelligence; it is a game of imperfect information.

Background and Motivation

Last year, DeepMind and Blizzard released the SC2 Learning Environment, an open source of tools used for AI research in StarCraft II. The release includes machine learning API, a python component of the environment (PySC2), and a large dataset of replays to analyze. The Starcraft II API provides full external control and includes functionality for machine-learning based bots and replay analysis. PySC2 provides an interface for reinforcement learning agents to interact with the game by fetching observations and sending actions.

StarCraft II presents many challenges as an AI environment because of its vast observation and action space. In the observation space, there are many structural and spatial elements to consider such as map terrain and the locations of resource clusters. However, SC2LE simplifies the game down into “feature layers,” which isolate elements of the game such as unit type, health and map visibility from one another, whilst preserving the core visual and spatial elements of the game. The action space is even larger, giving a player the choice to select actions among a combinatorial space of $10^8$ possibilities using Starcraft II’s point-and-click interface.
The large action space is a byproduct of the many different types of units and buildings, which each have their own unique set of actions. Since games typically last for thousands of frames, the consequences of a player’s actions may not be realized until much later in the game. The sheer magnitude of both the observation space and action space of *Starcraft II* has been a major obstacle for researchers interested in creating competitive *Starcraft II* agents. Through DeepMind’s AI, AlphaStar, it was only very recently (January 24, 2019) that computer scientists successfully constructed a competitive agent that beat professional players. Even AlphaStar, however, has its limitations. Since each race has its own unique set of features, AlphaStar was designed to only play PvP matchups. Nevertheless, the creation of AlphaStar was a major breakthrough in the computational intelligence for games and we anticipate that AlphaStar will eventually be able to competitively play *Starcraft II* in the other race matchups. Unfortunately, the time, extensive processing power, and resources the AlphaStar team had access to are not available to our team. Thus, our project aims to simplify the problem and attain a more modest goal: develop a program that predicts the win probability of a given *Starcraft II* player during a competitive 1v1 *Starcraft II* match.

The eSports industry is growing at an explosive rate, and with competitive professional matches comes pre- and live game betting. Being able to predict the outcomes of matches between players has become increasingly more important, and our program may have important applications in quantifying skill levels among eSports participants. Additionally, by training an AI agent to predict win probability rather than to competitively play the game, we eliminate some of the problems that negatively affect the game-playing bots. In terms of the observation space, the AI agent would have accessibility to all observable features of the game and would no longer be hindered by the “fog of war” game mechanic. Furthermore, our agent would no longer have to make decisions on actions to take, as it would not be playing the game, only observing it.

**Project Details**

The project would consist of three main parts. Because *StarCraft II* involves a large and complex environment with many inputs that requires preprocessing (like unit composition and positioning), the first part of the project will consist of getting familiar with the *StarCraft II* python environment and developing initial ideas on which inputs to use and how to use them in the neural net. For this part, I will work on coding a neural net using inputs from *microRTS*. *MicroRTS* is a simplified implementation of RTS games like *Starcraft II*, making it a good platform for us to develop and test our initial theories on predicting win probability in RTS games. I plan on building the neural net using complex data inputs that require preprocessing like unit positioning or unit composition that we may try to replicate in our *StarCraft II* model. Meanwhile, James will first familiarize himself with the SC2LE framework and work on constructing a basic neural network that uses basic in-game environment inputs not specific to race. Part II will consist of using our combined knowledge from *microRTS* and SC2LE to further develop the initial framework of our *Starcraft II* neural network for specific race matchups and training it using the 1v1 replay packs provided by SC2LE. Although every race has the same set of general inputs like supply or resources, each race also has its own unique set of inputs/attributes, making every matchup different in its own way. For example, all three races have their own set of units, buildings, and upgrades, and many units have abilities that are unique to its own race. For this part, I plan on using James’s initial general neural network framework and developing it specifically for the TvT matchup, using my *microRTS* neural net to help incorporate certain inputs from the SC2 environment that we did not utilize before. Multiple
frames from every replay will be taken as individual data points. The network will then be used to predict the outcome (or player’s win probability) of each of these frames. Part III will consist of analyzing the accuracy of our neural network and improving its accuracy by adding, subtracting, or changing input vectors. In order to assess the accuracy of our model’s win probability, we intend to assign scores to each data point prediction, rewarding the neural network if it predicts a winning player to have higher than a 50% win probability in the current game frame, and penalizing it if it predicts otherwise.

**Division of Work**

During the initial phases of the project, I will work on getting familiar with the *microRTS* game code and developing a neural network that implements basic win prediction. Simultaneously, James will construct the basic framework to analyze the replays of *Starcraft II* games and to access in-game data. After familiarizing ourselves with our respective games, I will work on preprocessing complex inputs in *microRTS* and utilizing them for my neural net, while James constructs a simplified neural network that disregards the race matchup to roughly estimate win probability.

While we anticipate working independently on these assignments, we also expect to communicate with one another in order to ultimately design a *Starcraft II* neural network that incorporates ideas garnered during the *microRTS* testing phase. After this, we would further develop and train our neural network on different race matchups. I would be working on the TvT matchup, while James works on the PvP matchup. During this we would cooperate to search for and develop potential useful inputs that require data cleaning and refining such as a player’s overall unit composition and structural positioning. We anticipate testing and refining various inputs to our then separate models, creating multiple iterations of our *Starcraft II* neural network. Our project would culminate in an analysis of how these inputs differ and how accurate each model is for our specified race matchup.

**List of Deliverables**

- Source code of both neural network agents (*microRTS* and *Starcraft II*)
- Final paper that details our methodology, the various neural networks tested, and their respective results
- Oral presentation to present our findings

**Timeline**

- **Week 1**: Refine and finalize research proposal with Professor Glenn.
- **Week 2-3**: Work on *microRTS* to develop a basic neural network that implements basic win prediction. (James: Get familiarized with PySC2 and code framework to analyze the replays of professional *Starcraft II* games).
- **Week 4-6**: Further develop *microRTS* neural network to handle more complex input. (James: develop initial neural network using non-specific race inputs).
- **Week 7-11**: Develop a *StarCraft II* neural network specifically for the TvT matchup and refine a list of significant inputs to be used using our *microRTS* neural network as a guide. Train the network using TvT ladder replays provided by SC2LE.
- **Week 12-14**: Analyze the accuracy of the model and observe how certain inputs affect its accuracy.
- **Week 15-17**: Synthesize all of the data and write a final research paper with our results