# Learning The World's Hardest Game

Yasyf Mohamedali

*6.867 Project Proposal*

**Abstract**

The World's Hardest Game is a Flash game made popular online in 2007. It features a simple square player, whose goal is to navigate a series of levels by traveling from a designated start area to a designated finish area, avoiding deterministically-moving enemies and collecting all coins on the map. While the game results in a rather frustrating experience for most human players, I propose teaching a computer to play (and master) the game, using Deep Q-learning with Experience Replay.

## 1. Introduction

The World's Hardest Game is an infuriatingly difficult yet simple Flash game which is widely available online [1]. In the game, a player selects a move from a discrete set (up, down, left, right, stay) to make at each time step, with the goal of navigating through a series of levels. In order to complete a level, the player must collect all on-screen coins, avoid collision with any enemies, and reach the area designated as the end zone. While the game is deterministic (modulo some random initialization of the enemies), it is a very difficult task for humans to complete, requiring precision, patience, and often a difficult-to-discover strategy for movement. However, the set of actions to be taken at any given time step is limited, and the reward for an action is easily calculated. Thus, the game is a good candidate for a reinforcement learning-based model.

[1] http://www.worldshardestgame.org/

## 2. Model

I propose building multiple models to compare performance in training a computer to play The World's Hardest Game, with the primary model being a Deep Q-learning model with Experience Replay, as was demonstrated in the Atari DeepMind paper [2]. Deep Q-learning takes the classic idea of Q-learning, which solves optimizes a MDP by calculating an optimal action-value function $Q$, and trains a deep neural net to act as a function approximator for $Q$. This model would use a CNN to first learn the state representation of the game from preprocessed images of simulated gameplay, and then the estimated optimal action values. In addition to this, I plan on exploring models that extract the game state from gameplay images in a deterministic fashion, using this as a direct input to a FCNN which learns $Q$. Finally, I will compare the two of these to a baseline which implements a random or greedy model.

## 3. Risks

There are several risks to this proposal. The most obvious is that the excellent results demonstrated by DeepMind on the Atari games will not be comparable to the performance on The World's Hardest Game. Indeed, the methodology that was used for the Atari games was created for more generic gameplay which is different from the game at hand. This may require further exploration of variant algorithms.

Other risks include the lengthy training time associated with image-based reinforcement learning, and the difficulty of simulating a Flash game in a stable, replicable manner. The training time issue can potentially be mitigated by using a pool of simulators which all reflect a certain history, such that there is always one ready to have a new action taken. This removes any additional time spent in re-creating states. Since the game is largely deterministic, caching states given a start state and history can also be explored. On the simulation difficulties front, I have already begin prototyping a simulator which can successfully model and capture the game frames.

[2] https://arxiv.org/abs/1312.5602

## 4. Plan

*Week 1*

In week 1, I will complete the simulator and ensure that game frames and states can be easily and efficiently captured and saved, so as to be able to generate the inputs to our model. I will also take this time to familiarize myself with the tools I will be using to implement this model, namely TensorFlow. Finally, I will take the time to come up with a viable reward function.

*Week 2*

Week 2 will see the start of my building the models in TensorFlow. I will begin with the baseline models, and ensure that the simulation-based data capture flow works flawlessly. I will implement the proposed models, and run some basic tests to ensure that they are easily trainable.

*Week 3*

The third week will have a majority of time spent on training and tuning hyperparameters, in order to achieve the best results. Any excess time will be used for experimentation in finding superior models.

*Week 4*

Finally, in the fourth week, I will spend time analyzing and summarizing my results. I will intentionally leave some of this time free as a buffer. If there is extra time, I intend to explore some of the more recent developments in reinforcement learning, such as Double Q-learning [3] or Dueling Network Architectures [4].

[3] https://arxiv.org/abs/1509.06461
[4] https://arxiv.org/abs/1511.06581