Project 1: AI Technique

By

Mr. WARIS DAMKHAM 6388014

Miss PATTANAN KORKIATTRAKOOL 6388022

Miss VIPAVEE NGAMYINGSURAT 6388094

Mr. THANAKIJ PINYOBOON 6388087

Submitted to

Dr. Thanapon Norase

Prof. Dr. Peter Haddawy

A Report Submitted in Partial Fulfillment of

the Requirements for

ITCS451_Artificial Intelligence

Faculty of Information and Communication Technology

Mahidol University

2021

Table of Contents

OpenAI Five with Dota2

On April 13th, 2019, OpenAI Five became the first AI system to defeat the world champions in an esports game. The game of Dota 2 presents novel challenges for AI systems such as long-time horizons, imperfect information, and complex, continuous state-action spaces, all of which will become increasingly central to more capable AI systems. OpenAI Five leveraged existing reinforcement learning techniques, scaling to learn from batches of approximately 2 million frames every 2 seconds. We developed a distributed training system and tools for continual training, which allowed us to train OpenAI Five for 10 months. By beating Team OG, the world champion of Dota 2, OpenAI Five shows that self-play reinforcement learning can help a computer do a hard task better than a human.

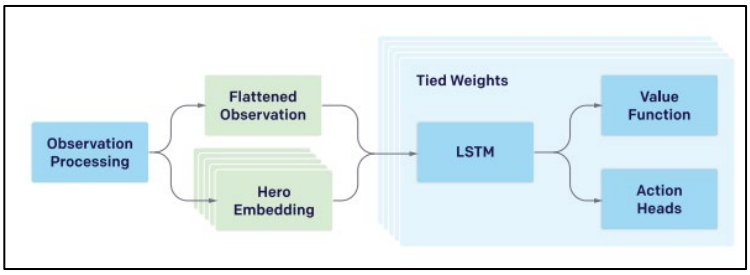
What is Dota 2?

Dota 2 is played on a square map with opposing bases defended by opposing teams. Each side's base has a building known as an ancient; the game finishes when the other team destroys one of these ancients. Each team consists of five players, each of whom controls a hero unit with distinct powers. During the course of the game, both teams have a steady stream of uncontrolled, little "creep" troops that move towards the opposing base and assault any opponent units or buildings. Players collect resources, such as gold, from enemies in order to boost their hero's strength by purchasing things and enhancing their powers.

What is OpenAI Five?

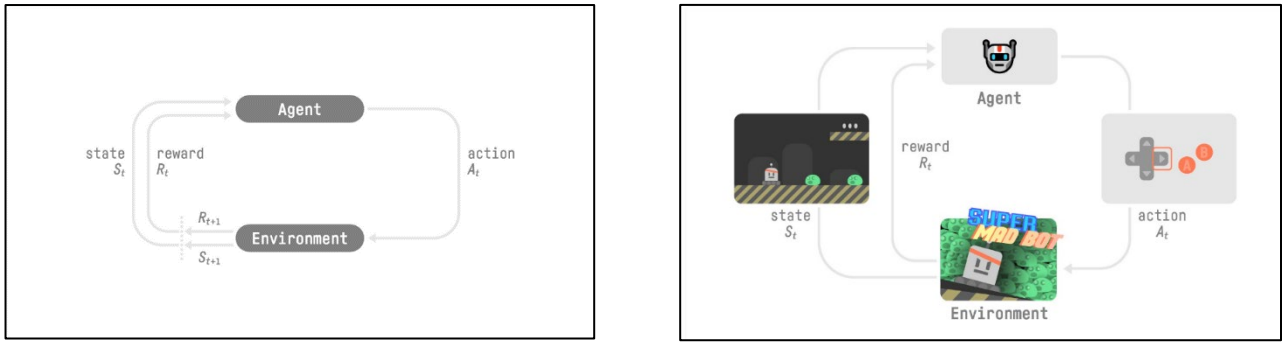
OpenAI Five plays 180 years' worth of games against itself every day, learning by self-play. It trains using a scaled-up version of Proximal Policy Optimization running on 256 GPUs and 128,000 CPU cores — a larger-scale version of the system we designed to play the much-simpler solo variation of the game last year. Using a distinct **LSTM** for each hero and no human data, it learns identifiable techniques. This indicates that **reinforcement learning** can yield long-term planning with large but achievable scale

Simplified OpenAI Five Model Architecture



The complicated multi-array observation space is transformed into a single vector before being put through a 4096-unit LSTM. The LSTM state is anticipated to achieve the policy outcomes (actions and value function). Each of the five heroes on the squad is controlled by a duplicate of this network with inputs that are virtually identical, but each has its own concealed state. Due to a portion of the observation processing output identifying which of the five heroes is being commanded, the networks conduct distinct actions. The LSTM accounts for 84% of the model's total number of parameters.

The Reinforcement Learning Framework



The RL Process: a loop of state, action, reward, and next state

What is LSTM Network?

Long Short-Term Memory networks (LSTMs) are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997) and were refined and popularized by many people in following work. LSTMs work tremendously well on a large variety of problems and are now widely used.

LSTMs are designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

What is Reinforcement learning?

Reinforcement Learning is based on the premise that an agent (artificial intelligence) would learn from its environment by interacting with it (through trial and error) and obtaining incentives (positive or negative) as feedback for executing actions.

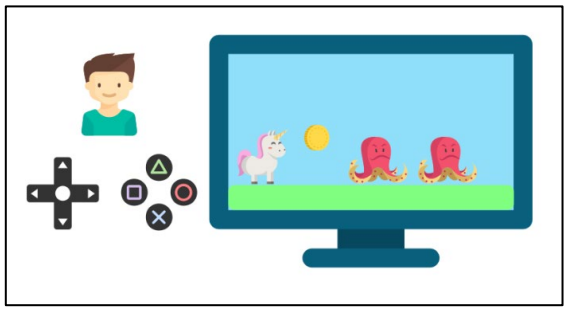


Fig 1

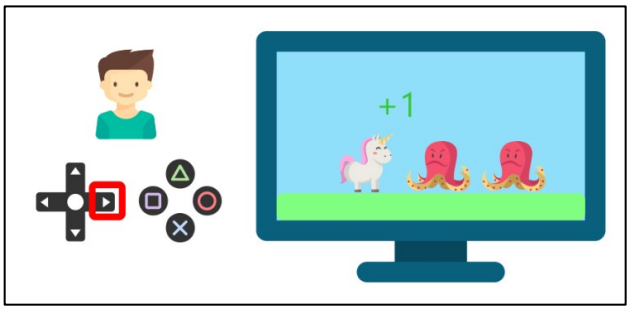


Fig 2

By clicking the correct button, Agent will interact with the surroundings (the video game) (action). He received a coin, which is a +1 reward. It's positive; he just realized that he must collect coins in this game. (Fig2) But then, he presses right again, and he touches an enemy, he just died -1 reward. (Fig 3)

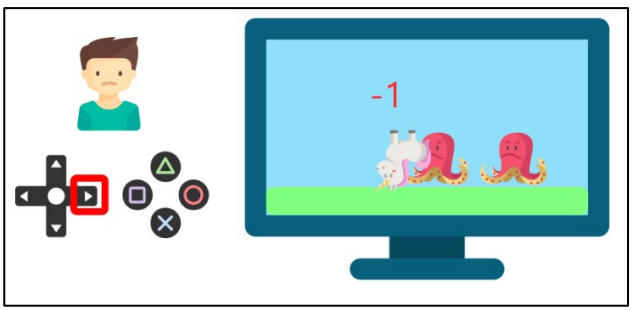


Fig 3

By interacting with his environment and learning via trial and error, your agent realized he needed to obtain money while avoiding adversaries in this setting. Without supervision, the agent will become increasingly proficient at playing the game. Interaction is how both people and animals acquire knowledge. Reinforcement Learning is just a computational method for learning from experience.

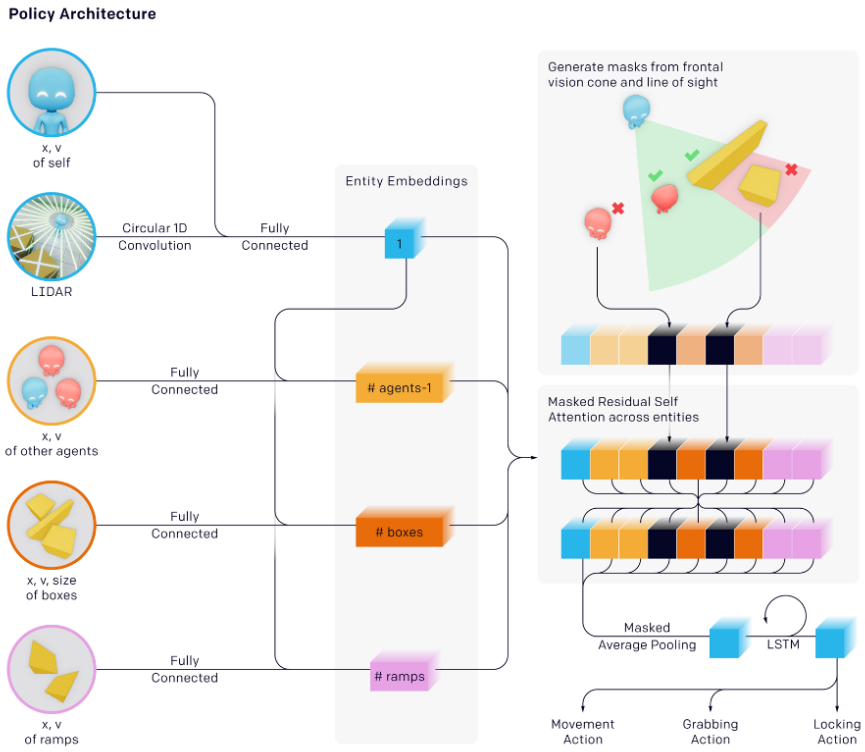
How are they different from the class content?

Using the minimax search technique in a competition is distinct from reinforcement learning (RL). Minimax search is based on competitive settings in which two players compete for the highest score. It may be comparable to a turn-based game in which one offensive player and one defensive player battle for the least number of points, such as chess, hex games, etc., in order to win. The algorithm for minimal search involves rounding. However, Reinforcement Learning (RL) is a computer method to action-based learning. We develop an agent that interacts with its surroundings through trial and error and receives negative or positive incentives as feedback. Any RL agent's objective is to maximize its predicted cumulative reward (also called expected return).

Example of game

Multi-agent Hide and Seek from OpenAI

Multi-agent Hide and Seek uses the same training infrastructure and algorithms used to train OpenAI Five.



Each agent in our environment behaves autonomously based on its own observations and concealed memory state. Agents adopt a permutation-invariant entity-centric state-based representation of the world with regard to objects and other agents. Similar to transformers, each item is embedded and then processed via a masked residual attention block. Objects not in the agent's line of sight and in front of the agent are disguised so that the agent is unaware of their existence.

References

OpenAI. 2022. *OpenAI Five*. [online] Available at: https://openai.com/blog/openai-five/

Colah.github.io. 2022. *Understanding LSTM Networks -- colah's blog*. [online] Available at: http://colah.github.io/posts/2015-08-Understanding-LSTMs/#lstm-networks

Huggingface.co. 2022. *An Introduction to Deep Reinforcement Learning*. [online] Available at: https://huggingface.co/blog/deep-rl-intro#the-reinforcement-learning-framework

Arxiv.org. 2022. [online] Available at: https://arxiv.org/pdf/1912.06680.pdf

OpenAI. 2022. *Emergent Tool Use from Multi-Agent Interaction*. [online] Available at: https://openai.com/blog/emergent-tool-use/

Poker: Libratus vs Four top player

The story of Poker: Libratus vs Four top player

- Libratus is an artificial intelligence designed to play poker. The creators of Libratus intended to make Libratus universally applicable to applications other than poker. Carnegie Mellon University developed Libratus, later Libratus beat four of the best heads-up no-limit “Texas hold’em” poker players in the world in a tournament, 20-day competition.

Al technique

- **Nash equilibrium**

  Nash equilibrium is a concept of decision-making theorem within game theory where the optimal outcome of a game is where there is no incentive to deviate from the initial strategy. Each player's approach in the Nash equilibrium is the best one given what the other players have decided. Every player receives the result they want, and every player wins. In this case, AI is trying to find a rough approximation for how to play the entire game before the competition begins. Then AI is planning to win against humans by trying to get itself into a situation with a better strategy.

- **Optimal Decision**

  Libratus is a scenario where multiple decision makers compete under imperfect information. It specifically takes into account the fact that in a poker game, both players are attempting to maximize their own interests in order to deal with the unusual combination of many agents and imperfect information.

- **Zero-sum game**

  Even though Nash equilibrium is an immensely important notion in game theory, it’s not a specific gameplay strategy, so it is hard to clarify which one is the perfect optimal decision. Poker is one of several two-player games that has the extra feature that one player's prize is the opposite of their opponent's. Zero-sum games are one type of such game. Regardless, in theory, the Max agent and Min agent both want to compete to win. So both agents get opposite outcomes, while the agent ‘Max’ is trying to maximize the winning score, the agent ‘Min’ also tries to minimize the winning score as well.

- **Monte Carlo's counterfactual regret minimization**

  While playing the game in real-time, AI is trying to precompute a solution to streamline abstracting by using Monte Carlo's counterfactual regret minimization algorithm. Counterfactual regret minimization is the iterative algorithm, one player is going to traverse the game tree by playing at random and update the regrets while other players are going to sample actions. The value of regrets will correspond to how much players 'regret' having not taken an action in the previous iteration then there will be a 'regret' value for each action which the player can take at every decision-making point

Compare and contrast: How they are different from the class content

- The Poker: Libratus vs Four top player has used the Nash equilibrium and Monte Carlo's counterfactual regret minimization algorithm for AI to do the decision-making and to compete to win which is different from the class content of ITCS451 class. In the class, the lesson that has been taught is the Minimax algorithm. The difference between Monte Carlo's algorithm and Minimax algorithm is that the Minimax algorithm is a recursive algorithm which is used in decision-making in game theory. It provides an optimal move for the player assuming that the opponent is also playing optimally. While Monte Carlo's algorithm is the iterative algorithm, one player is going to traverse the game tree by playing at random and update the regrets while other players are going to sample actions.

Apply the AI techniques to other game

- The AI techniques of the 'Libratus' can be applied to other games such as Checkers, tic-tac-toe and other two-players games.

Reference

[1] https://www.youtube.com/watch?v=2dX0lwaQRX0

[2]https://www.cio.com/article/234157/in-major-ai-win-libratus-beats-four-top-poker-pros.html

[3] https://thegradient.pub/libratus-poker/

[4] https://www.engadget.com/2017-02-10-libratus-ai-poker-winner.html

Chess IBM deep blue vs Players

Overview

IBM is the company that created the deep blue algorithm. It was a chess-playing computer system made to compete against the current world champion in a standard chess game or match within certain time constraints.

The massive parallel system known as Deep Blue is used to do chess game tree searches, which show every potential move from the game's beginning in graphical form. The system was created using an IBM RS/6000 SP computer with 30 nodes (30 processors) and single-chip chess search engines with 16 chess chips per SP processor. There were 2 nodes with 135 MHz P2SC processors and 28 nodes with MHz P2SC processors in the SP system. Each node has 1 GB of RAM and 4 GB of hard drive space, and they all connect to one another by a fast switch. Deep Blue's chess chips can each search between 2 and 2.5 million chess positions per second. They use a microchannel bus to connect with their host in order to accomplish that.

AI Techniques

- **Tree Search**

Chess is fundamentally modeled as a Tree Search problem, where each state represents a specific configuration of the pieces on the board and the available actions represent the permitted movements for the current player in that configuration.

After modeling the game in this fashion, we can start using the algorithms we learned in this course to solve the issue.

- **The Evaluation Function**

The evaluation function is the main component of a chess-playing program, according to Shannon's article. We must develop a function that takes in a game state (in our case, a board configuration) and reduces it to a real-number evaluation of the state because we can't predict whether a particular move will win all the way up until the end of the game (especially since we don't know what the other player will do during their turns!). Instances of the board where the player of interest has more of their pieces on the board than the rival could result in greater scores from the function. We would likely like the function to assign a very high score in

particular. (Perhaps even infinite) to the board configuration when the opponent's king is in checkmate, ensuring the game victory for the player of interest.

- **The Minimax Algorithm**

All that remains is a method for actually selecting which action to do after an examination. Even if choosing the move that results in the board arrangement with the highest evaluation score by merely thinking one step ahead would be a solid starting point, we may be even smarter and consider the potential moves our opponent might make after we have moved. This insight gives rise to the "Minimax algorithm," so named because we select the course of action that minimizes the maximum "loss" we could incur from doing a specific action. In particular, after making our initial move, we assess all the potential actions our opponent might make in each of their subsequent turns by looking as far ahead as our processing capability would allow. The move we may make that would minimize this maximum "loss" (or minimum of our evaluation function) that our opponent could cause for us through their movements is then selected.

- **Heuristics/Optimizations**

One may already create an immensely powerful chess-playing program with just an evaluation function and an implementation of the minimax algorithm. However, given the unique characteristics of chess, the "large time" systems go much further than this by incorporating "heuristics," or quick rules that can reduce computing time, as well as minimax algorithm optimizations. Alpha-beta pruning is a well-known minimax optimization in which any move for which another move has previously been found that is certain to perform better than it is deleted.

Opinion

We discovered, among other things, that there are various angles from which to view complex issues. For instance, there are two ways to play chess: the human approach, which relies heavily on pattern recognition and intuition, and the machine way, which relies heavily on searching through millions or billions of possibilities. These methods frequently complement one another.

In many real-world issues, computers and humans working together perform better than each one working alone. This is unquestionably true in the game of chess. Because diagnosing a patient involves many intangibles that are challenging to capture in statistics, we wouldn't want,

for instance, computers to take over patient diagnosis and treatment on their own. But a system like that can be quite helpful for recommending alternatives to consider possibly those that are from very recent technical articles or clinical trials that the doctor may not be aware of.

Reference

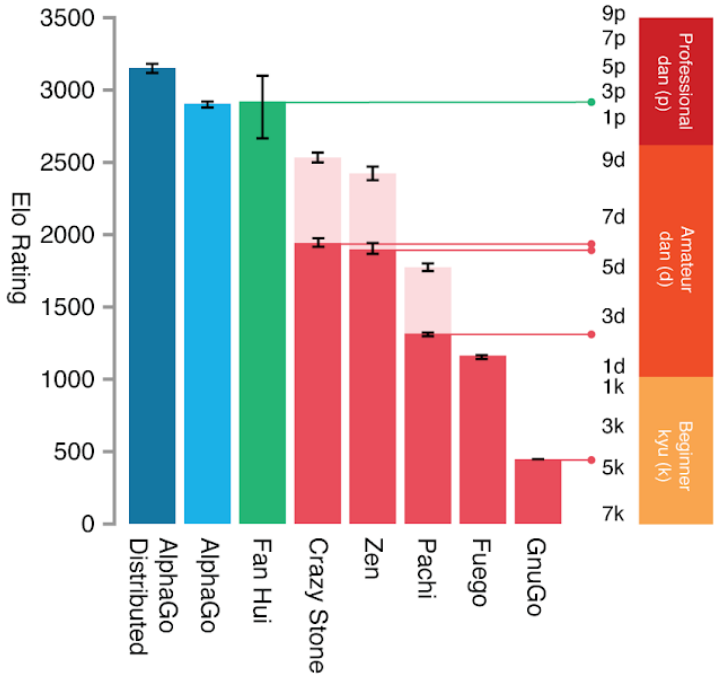
CS221-deepBlue | Stanford.edu url: https://stanford.edu/~cpiech/cs221/apps/deepBlue.html

Deep Blue Algorithm: A Detailed Guide | Professional-ai.com URL https://www.professional-ai.com/deep-blue-algorithm.html

Go: AlphaGo vs Players

Go is one of the games with a high level of complexity. And has a long history of more than 2,500 years, currently with approximately 40 million players around the world. In the past, the development of Artificial Intelligence (AI) or artificial intelligence that can play Go. Still progressing to just amateur play only. Unlike chess that IBM has developed, AI has the ability to compete with world champions.

As a result, Google has begun to develop AI for chess with the idea of creating two Deep Neural Networks based on learning. Know millions of data The first set is called Policy Network for predicting the next turn of the opposing player. To reduce processing to the extent necessary to think in order to win. And recommend the best walking eyes while the other, known as the Value Network, is used to reduce the depth of the Search Tree and assess its current advantages and disadvantages.



This figure from the Nature article shows the Elo rating and approximate rank of AlphaGo (both single machine and distributed versions), the European champion Fan Hui (a professional 2-dan), and the strongest other Go programs, evaluated over thousands of games. Pale pink bars show the performance of other programs when given a four move headstart.

This concept has greatly reduced the amount of processing during board game play from full-blown thinking. By training its AI with more than 30 million eyes, the AlphaGo has become more proficient at predicting. The next turn of the human player is 57% accurate, which is 44% better than other AI. Then Google developed AlphaGo to learn and create new moves or strategies. Come out on your own from thousands of races against yourself to test a new kind of Trial-and-Error called Reinforcement Learning.

Of course, training AI at this level requires a lot of computing power, and Google the Cloud Platform is used for this purpose, with many CPUs and GPUs being used to process open-source deep learning systems like TensorFlow.

The results of this AI training were impressive, with AlphaGo defeating other AIs 499 times, losing only 1 time, with each match being preceded by another AI by placing checkers in the corners of the board. before reaching 4 corners together before the start of the game.

Compared to that, Deep Blue was able to defeat a chess champion. As you can see, the idea behind the development is different, as Deep Blue has a game orientation from a chess expert, but AlphaGo learns from the records of other pro players. As well as practicing by playing with yourself,

Deep Blue uses a search method for all eyes, but AlphaGo only searches part of all eyes. Then choose the eye that you think is the best from that section. It shows that the amount of walking calculation in each eye is significantly less than Deep Blue (200,000,000 vs 60,000).

Reference

[1] https://ai.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html

[2] https://www.wired.com/story/this-more-powerful-version-of-alphago-learns-on-its-own/

[3] https://www.wired.com/2016/05/google-alpha-go-ai/