Title slide

Motivation slide

Brief summary of previous work / background slide(s)

Topic summary covering key ideas slides

References (not included in the <= 20slide limit)

Limit the number of equations to just the key ones and maximize the number of figures and plots used to illustrate the key ideas. Text should come from you, not be copied from others. If you copy a figure from somewhere else, provide a reference.

Games / StarCraftII And Or Dota2

* Real-time strategy (RTS) games are recognized for their game theoretic and domain complexities. Many sub-problems of RTS games, for example, micromanagement, base economy, or build order optimization, have been studied in depth often in small-scale environments. For the combined challenge, the StarCraft domain has emerged by consensus as a research focus.
* AlphaStar uses a model-free, end-to-end learning approach to playing StarCraft II that sidesteps the difficulties of search-based methods that result from imperfect models, and is applicable to any domain that shares some of the challenges present in StarCraft. Dota 2 is a modern competitive team game that shares some complexities of RTS games such as StarCraft (including imperfect information and large time horizons) However, unlike AlphaStar, some game rules were simplified, players were restricted to a subset of heroes, agents used hard-coded sub-systems for certain aspects of the game, and agents did not limit their perception to a camera view.
* StarCraft II or Dota2 – testbeds to demonstrate how AI can perform on a larger scale better then even the world champions.
* These are meant to then translate their knowledge to other tasks than gaming to ease human work.
* (Google) StartCraft2 or (Openmind) Dota2 – Learn on previous gameplays
* Google Deepmind Atari 57 – Learn from the pixels information
* Nvidia Pacman – Look from the graphics (120 hours playing) and develop a similar looking game

|  |  |  |  |
| --- | --- | --- | --- |
|  | Atari | Go | Starcraft/Dota2 |
| Information Type | Near Perfect | Perfect | Imperfect |
| Players | Single Player | Multi-player | Multi-player |
| Action Space | 17 | 361 | ~10^26 |
| Moves per game | 100’s | 100’s | 1000’s |
| Additional aspects |  | Intuition | Intuition, team-work and collaboration |

**OpenAI Five Beats World Champion DOTA2 Team 2-0**

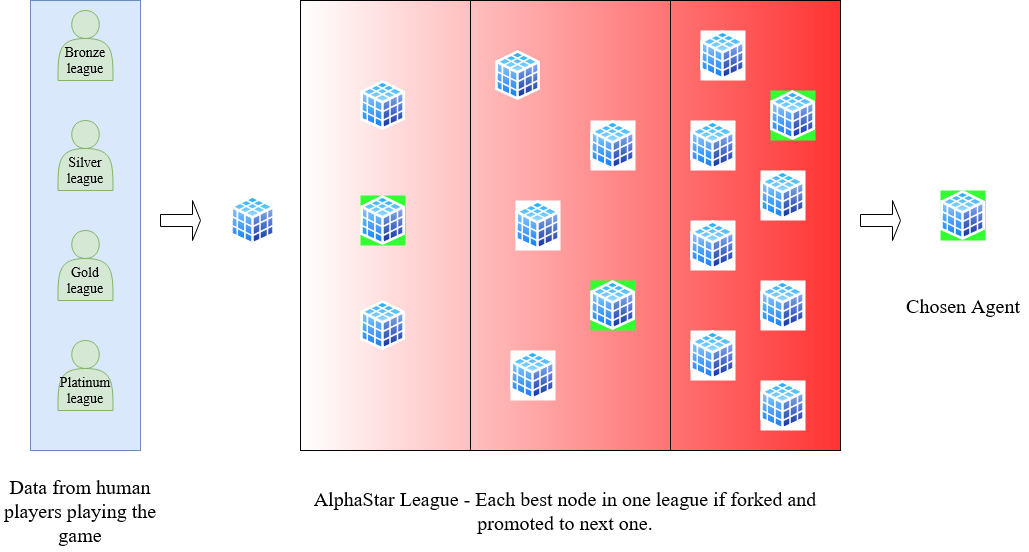
[**#Ref1**](https://www.youtube.com/watch?v=tfb6aEUMC04)**,** [**#Ref2**](https://openai.com/blog/openai-five-finals/)**,** [**#Ref3**](https://arxiv.org/abs/1912.06680)

* DOTA 2 - an esports game – defeat the opposite team and take on their buildings.
* Every hero has a variety of unique spells in a mirror made where both teams have the same heroes.
* Complex multiplayer real-time strategy games defeat top tier game players which require collaboration and teamwork.
* Rules of Dota 2 are so complicated that you cannot pre-program these rules. Even if you try to pre-program it, the end result would not be as good as a average player.
* Difficult because:
  + Requires teamwork and planning. Even one bad move and thousands good move after can result in losing the game.
  + Unknown imperfect environment giving partial observability – Can just know your own units(heroes), buildings can see.
  + Too much information and too many decisions to take – complex continuous state-action spaces.
* Long term strategic planning – can be a nightmare for an AI to look forward – Use Reinforcement learning on a 100,000 CPU and bots learn from every game they play. – Self play algorithm – trained of lifetime of games, explored so many strategies – 2 million frames per 2 seconds.
* Although the Dota 2 engine runs at 30 frames per second, OpenAI Five only acts on every 4th frame which we call a timestep. Each timestep, OpenAI Five receives an observation from the game engine encoding all the information a human player would see such as units’ health, position, etc. OpenAI Five then returns a discreteactionto the game engine, encoding a desired movement, attack, etc.
* Certain game mechanics were controlled by hand-scripted logic rather than the policy: the order in which heroes purchase items and abilities, control of the unique courier unit, and which items heroes keep in reserve. While we believe the agent could ultimately perform better if these actions were not scripted, we achieved superhuman performance before doing so.
* OpenAI Five plays 180 years worth of games against itself every day, learning via self-play. It trains using a scaled-up version of [Proximal Policy Optimization](https://blog.openai.com/openai-baselines-ppo/) running on 256 GPUs and 128,000 CPU cores. Using a separate [LSTM](http://colah.github.io/posts/2015-08-Understanding-LSTMs/#lstm-networks) for each hero and no human data, it learns recognizable strategies. This indicates that [reinforcement learning](https://www.technologyreview.com/s/603501/10-breakthrough-technologies-2017-reinforcement-learning/) can yield long-term planning with large but achievable scale — without fundamental advances, contrary to our own expectations upon starting the project.
* Utilized a distributed training system to do this over a period of 10 months utilizing thousands of GPUs.
* Training over months – will have changes – on which we need to restart training – need better solution – Surgery
* Drawback of self-play is that they forget how to win against previous versions of themselves as they improve.
* The algorithm passes through a Vision network which then passes through an imitation network.
* Tuning of hyperparameter called – ‘team spirit’ – bots are selfish – tuning this parameter enables them to care about their team members.
* In 2017, Open AI to beat world champion an DOTA 2 with one hero.
* In 2018, Open AI5 challenged the world champion team OG for a 3-round series.
* The AI does not look at game as the pixel information we see, but as bunch of numbers.
* Training – Played a pool of 17 heroes over 10,000 years of games against itself.
* Testing – AI plays unusually aggressively and use buybacks quite liberally where human players don’t consider it to be a good choice. (buybacks – resurrect a perished hero quickly costing some money). The AI knows that if the game goes on for a long time, statistically their chances to win the game dwindles, so it immediately need to go and win from the start whatever the cost.
* AI knows when to engage and when to backout with a slight life left of the hero. It can assess whether a fight is worth pursuing. Or if said other way round, if an AI is fighting you, it certainly is having a high chance of winning.
* In a unprecedented move by human player, it kills 2 players of the opposite team before being killed by the human player team, almost like a bait to come out favorably. It’s doing things that you’ve never done and you’ve never seen. Similar behavior was seen in the series between Alpha Go and Google’s Deepmind where the machine won 4-1 against Lee Seedol (World champion of Go). It is said that the number of possibilities to play in a game of Go is more than the atoms in the world and there is no brute force way to get around it even with the existing computing power. In the game, some non-anticipated moves were played where the machine emphasized on winning the game by taking slack(lazy/not preferred) moves, since it has to win even if it by a very small margin.
* AI waited for the right moment to win, in another it pressured the human player from the starting and never let the human player reach the end game to execute their strategies.
* Humans can definitely use AI to understand why a weird move was played if it generated a better outcome than they thought and use it to improve their own game.
* One key learning that we took is how it was allocating resources. It’s just allocating resources as efficiently as possible. And then you realize that we’re guilty of being stuck in a team dynamic, whereas sometimes we have to be way more flexible.
* In an internet scale Arena OpenAI challenged and won with an 99.4%-win rate (7215 wins, 42 losses) for 33.7K players.

# DeepMind’s Alpha Star Beats Humans 10-0 (or 1)

[**#Ref1**](https://www.youtube.com/watch?v=jtlrWblOyP4)**,** [**#Ref2**](https://www.nature.com/articles/s41586-019-1724-z.epdf?author_access_token=lZH3nqPYtWJXfDA10W0CNNRgN0jAjWel9jnR3ZoTv0PSZcPzJFGNAZhOlk4deBCKzKm70KfinloafEF1bCCXL6IIHHgKaDkaTkBcTEv7aT-wqDoG1VeO9-wO3GEoAMF9bAOt7mJ0RWQnRVMbyfgH9A%3D%3D)

* StarCraft is a real-time strategy game in which players balance high-level economic decisions with individual control of hundreds of units.
* In StarCraft, each player chooses one of three races—Terran, Protoss or Zerg—each with distinct mechanics. We trained the league using three main agents (one for each StarCraft race), three main exploiter agents (one for each race), and six league exploiter agents (two for each race).
* Each player starts with a number of worker units, which gather basic resources to build more units and structures and create new technologies. These in turn allow a player to harvest other resources, build more sophisticated bases and structures, and develop new capabilities that can be used to outwit the opponent. To win, a player must carefully balance big-picture management of their economy - known as macro - along with low-level control of their individual units - known as micro.
* Challenges:
  + **Exploration-Exploitation Balance:** In StarCraft II there is no single winning strategy. At any given time, the AI agent needs to balance the need of exploring the environment in order to expand its strategic knowledge instead of taking actions that can yield immediate benefits.
  + **Imperfect Information:** Unlike games like chess in which players can observe the entire environment, StarCraft II never presents the complete environment configuration at any given time. From that perspective, an AI agent needs to be able to operate using imperfect information.
  + **Long-Term Planning:** A typical StarCraft II game takes about 1 hour to complete and, during that time, players are constantly taking actions to execute on an overall strategy. Actions that are taken early in the game might not take effect until much more later which require constant long-term planning abilities.
  + **Real Time:** One thing is strategic planning and another one is real time strategic planning. In classic chess, players can safely take 1 hour to evaluate a single more but, in StarCraft II actions need to be taken real time. From the AI perspective, this means that agents need to evaluate thousands of options real time and detect the best match for the long term strategy.
  + **Large Action Space:**  The StarCraft II environment requires players to control hundreds of units at any given time and the combinatorial combinations of actions grow proportional to the complexity to the environment.
* At each step t, our agent AlphaStar receives an observation that includes a list of all observable units and their attributes. (the game includes only opponent units seen by the player’s own units, and excludes some opponent unit attributes outside the camera view)
* Each action is highly structured: it selects what action type, out of several hundred (for example, move or build worker); who to issue that action to, for any subset of the agent’s units; where to target, among locations on the map or units within the camera view; and when to observe and act next.
* AlphaStar is trained via both supervised learning and reinforcement learning. In supervised learning, the parameters are updated to optimize Kullback–Leibler (KL) divergence between its output and human actions sampled from a collection of replays. Updates are applied using Adam Optimizer along with L2 regularization. In reinforcement learning, human data are used to sample the statistic z, and agent experience is collected to update the policy and value outputs via reinforcement learning (TD(λ), V-trace, UPGO) combined with a KL loss towards the supervised agent. We use the match out - come (−1 on a loss, 0 on a draw and +1 on a win) as the terminal reward , without a discount to accurately reflect the true goal of winning games.
* Central to AlphaStar is a policy represented by a neural network with parameters that receives all observations from the start of the game as inputs and selects actions as outputs. The policy is also conditioned on a statistic z that summarizes a strategy sampled from human data.
* Three pools of agents, each initialized by supervised learning, were subsequently trained with reinforcement learning. As they train, these agents intermittently add copies of themselves—‘players’ that are frozen at a specific point—to the league. The main agents train against all of these past players, as well as themselves. The league exploiters train against all past players. The main exploiters train against the main agents. The parameters of the agent are updated from the outcomes of those games by the actor–critic reinforcement learning procedures. Main exploiters and league exploiters can be reset to the supervised agent when they add a player to the league.
* AlphaStar uses a combination of new and existing general-purpose techniques for neural network architectures, imitation learning, reinforcement learning, and multi-agent learning.
* Observations of player and opponent units are processed using a self-attention mechanism.
* To integrate spatial and non-spatial information, we introduce scatter connections. To deal with partial observability, the temporal sequence of observations is processed by a deep long short-term memory (LSTM) system.
* To manage the structured, combinatorial action space, the agent uses an auto-regressive policy and recurrent pointer network.
* Supervised learning is performed on a dataset of 971,000 replays played on StarCraft II versions 4.8.2 to 4.8.6 by players with MMR scores (Blizzard’s metric, similar to Elo) greater than 3,500, that is, from the top 22% of players.
* From each replay, we extract a statistic z that encodes each player’s build order, defined as the first 20 constructed buildings and units, and cumulative statistics, defined as the units, buildings, effects, and upgrades that were present during a game. We condition the policy on z in both supervised and reinforcement learning, and in supervised learning we set z to zero 10% of the time.
* After supervised learning, the agent parameters were subsequently trained by a reinforcement learning algorithm that is designed to maximize the win-rate (that is, compute a best response) against a mixture of opponents. The opponent is decided by a multi-agent procedure.
* AlphaStar’s reinforcement learning algorithm is based on a policy gradient algorithm similar to advantage actor–critic. Updates were applied asynchronously on replayed experiences. In large action spaces, previous and current policies are highly unlikely to match. This requires an approach known as off-policy learning, that is, updating the current policy from experience generated by a previous policy.
* The league consists of three distinct types of agent, differing primarily in their mechanism for selecting the opponent mixture. First, the main agents utilize a prioritized fictitious self-play (PFSP) mechanism that adapts the mixture probabilities proportionally to the win rate of each opponent against the agent; this provides our agent with more opportunities to overcome the most problematic opponents.
* Second, main exploiter agents play only against the current iteration of main agents. Their purpose is to identify potential exploits in the main agents; the main agents are thereby encouraged to address their weaknesses. Third, league exploiter agents use a similar PFSP mechanism to the main agents but are not targeted by main exploiter agents. Their purpose is to find systemic weaknesses of the entire league. Both main exploiters and league exploiters are periodically reinitialized to encourage more diversity and may rapidly discover specialist strategies that are not necessarily robust against exploitation.
* For every training agent in the league, we run 16,000 concurrent StarCraft II matches and 16 actor tasks (each using a TPU v3 device with eight TPU cores23) to perform inference.
* Require split second decision making, it has imperfect environment, the AI only sees what its units can see.
* The training time for the initial version was about 2 weeks.
* This also used self-play algorithms against itself. DeepMind team inserted exploiter units in their game to balance their AI to not over fit the game plays.
* Building blocks of Deep mind StarCraft (Alpha Star) were reasonably general and are meant to apply the similar algorithms to various tasks and learn by itself like weather prediction and climate modeling.
* The Actions per minute(APM) and the response time(350ms) was set to be of an average player of the game, to be on a fair stage with the human player. This was made to ensure that the AI does not win by insanely clicking during the map, but by utilizing strategies.



Recently Deepmind worked on Deep Reinforcement learning technique to train a superhuman AI for 57 Atari Games which exceeded at human play in each of the game with a single generalized algorithm.

This AI was given a meta-controller which can decide on the short and long term planning.