**Evolution of Artificial Intelligence**

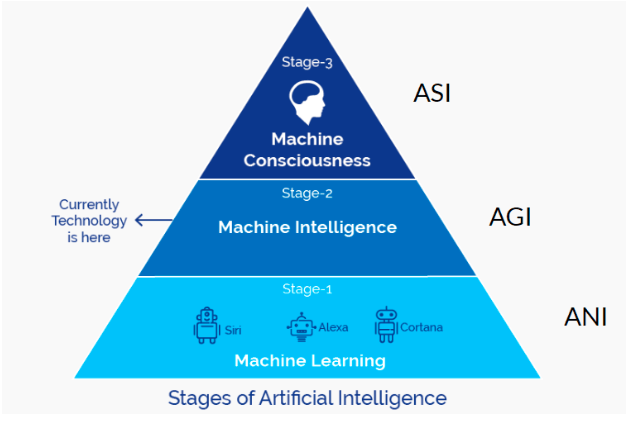
**Self-Play and Multi-Agent Interactions**

**Introduction**

In this blog, I will be talking about OpenAI’s strive for Artificial General Intelligence (AGI) and how they are progressing this topic by implementing their AGI in different environments with Self-Play and Multi-Agent Interactions.

**Different AI**

AI’s can generally be split into three sub categories: Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI), and Artificial Super Intelligence (ASI). Below is an illustration of the three types of AI and where we’re currently at.



The first stage ANI, as the name suggest is an AI with a limited scope of functionality. This is akin to simple machine learning model that predicts a target variable (such as regression). Although these models are able to automate a lot of the work and is able to achieve similar or better accuracy than us humans, any small tweaks in the environment (things like data, parameters, objective etc.) would most likely result in the model failing. This is because ANI are extremely sensitive to how it was trained, and is only able to do what it was programmed to do. An ANI that predicts the weather may be trained to predict weather in different regions based on the data we present to it. However if we change the objective (say the brand of cars), the model will not be able to adapt since it was not programmed to do so.

The second stage AGI is what we commonly associate with human-level AI. The defining feature of AGI is that it is able to learn in a completely different environment with very limited data and supervision. An AGI is able to perform the task even if the objective or environment is different (An example will be explored later on). In theory an AGI should be able to function on the same level as a human (i.e. able to do anything a human can), much like the robot from the movie “I-Robot”. Although we have yet to fully understand and achieve AGI, researchers have been able to create “semi-AGI” in the sense that the AI is able adapt to different environment without ever seeing it.

The last stage ASI is commonly known as Machine Consciousness, something that we have yet to come close to and is purely theoretical. As you’ve probably guessed from the previous stages, ASI in theory should be able to outperform a human in every way imaginable. ASI is not just mimicking a human, but is fully aware and can express cognitive thoughts such as feeling, empathy and more. In essence the theory of ASI is the point in which AIs not only think like humans but excel at everything that a human can.

Since ASI is COMPLETELY theoretical at the current time we will focus on ANI and AGI (even AGI is incomplete at the current time), more specifically how OpenAI is able show signs of AGI in games and simulations.

**Self-Play**

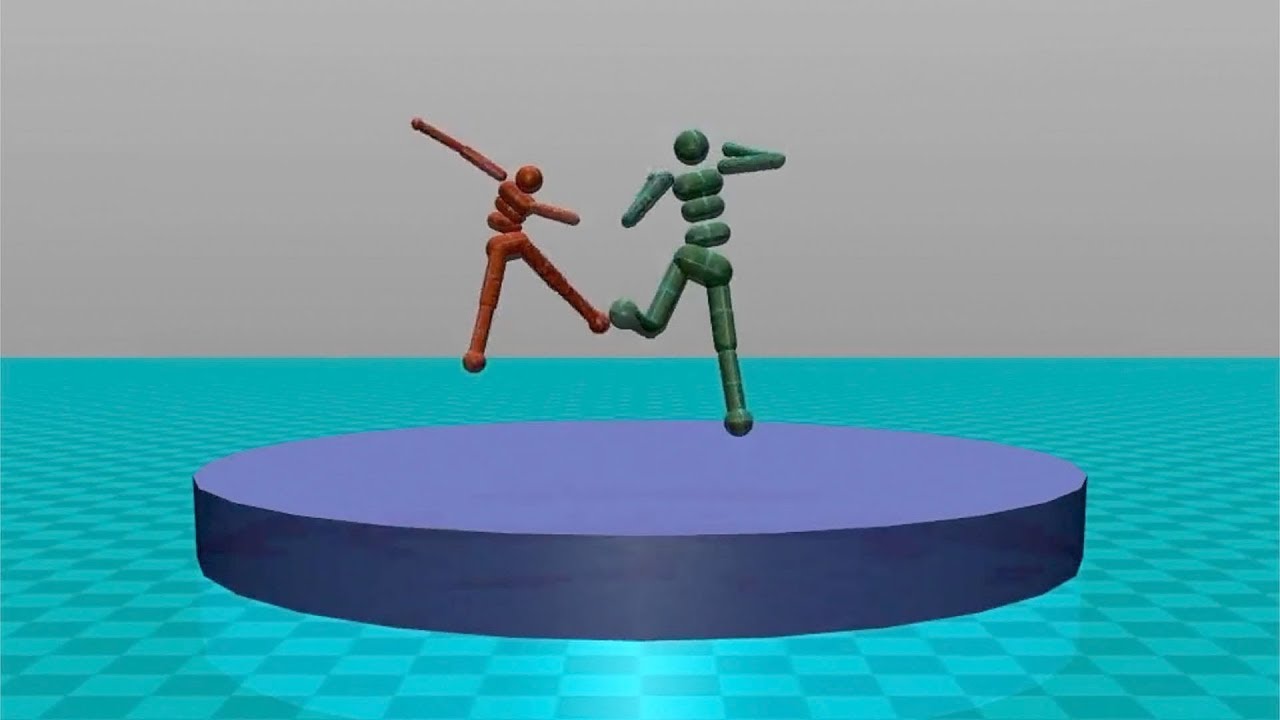
Self-play is an algorithm, or more specifically a reinforcement learning algorithm. Self-Play trains the agents (AIs) by making it play against itself continuously. The environment is set up with nothing but the basics (things such as rules and objective), and the agent is left to learn by itself via Self-Play. This is crucial because this means the AI is not learning off data provided by humans which could affect the learning. The technical side of the Self-Play algorithm is extremely complicated and to my knowledge is still not fully understood even by researchers. Hence I will try to explain how Self-Play works through examples and comparisons.

First imagine a simple case where I want to design and AI that can play Tic-Tac-Toe with the objective of winning (or Tie). Tie here counts as a win because Tic-Tac-Toe is a zero sum game; meaning if both players are playing with optimal strategy, the game will always end in a tie. This problem could be solved using “**Minimax”** algorithm. Minimax is an algorithm in which every possible state of the game is computed, then the machine determines the next move based on the board (grid) configuration and which move will result in the best outcome. This method is able to solve this problem exactly since the grid is so small. However if we extend this problem to board games like shogi, chess and Go, the Minimax would fail, simply because of how much more complex these games are. In cases like these algorithms such as “**Monte-Carlo tree search”** (<https://en.wikipedia.org/wiki/Monte_Carlo_tree_search>) or **“Alpha-beta Pruning”** (<https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning>) are used to approximate the Minimax algorithm.

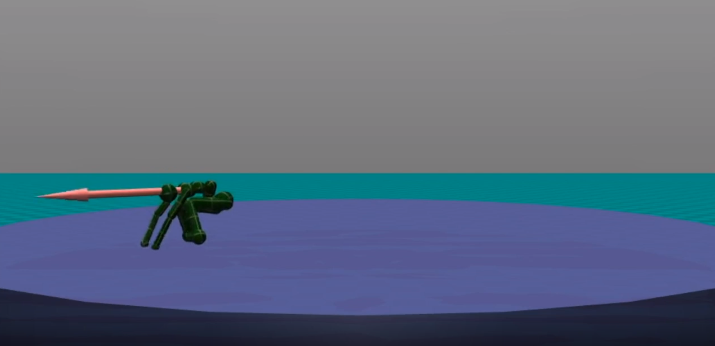
Deep Mind’s AlphaGo and AlphaGoZero were able to solve this problem using Self-Play. AlphaGo is the predecessor of AlphaGoZero, with the key difference of AlphaGo learning off games played by human pros whereas AlphaGoZero is learning by playing against itself. As I mentioned before, how data is presented to our model can greatly affect the output. “Garbage in Garbage out”, the model is only as good as our data allows it to be. AlphaGoZero was able to beat its predecessor 100 to 0, since AlphaGo is learning from pro player games, it will only ever be as good as the Pros who played those game, whereas AlphaGoZero have no such limitations. The only inputs given to AlphaGoZero are rules of the game and nothing else.

A general explanation of Self-play is essentially a single neural network at current time (T = t) playing against its former self at (t = t – 1), and between each step it uses algorithms such as Monte-Carlo tree search to update its policies (probability of available moves) and then updating the NN (e.g. new self).

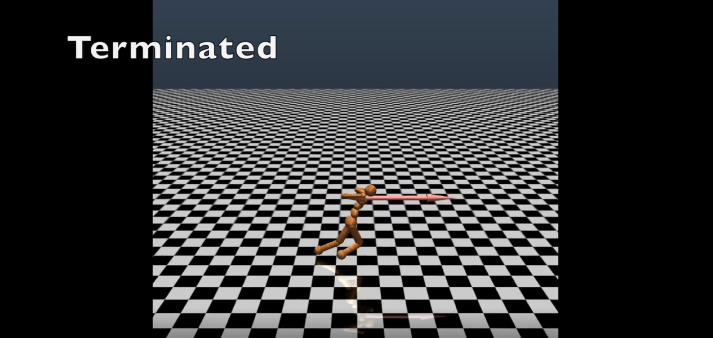
**Transfer Learning**

OpenAI utilized the same technique except in a Sumo game environment where agents are given task such as knock the other agent out of the ring, walk, run, push and various other behaviors. Agents were rewarded for exploring actions such as moving and standing, and this reward gradually adjusted towards zero and changed to reward only if the agent accomplishes its task. The AI bootstraps samples from its past selves only taking those that performed well and learning off them, thus creating AI that can perform better than humans (e.g. DOTA 2).

After playing against versions of itself multiple times it was introduced to a new environment with a new task. This new environment introduced wind and took away the opposing agent, as well as changed the objective to just staying in the ring. Even though the Self-Play Sumo AI has never seen this environment it was able to adapt and stay in the ring.



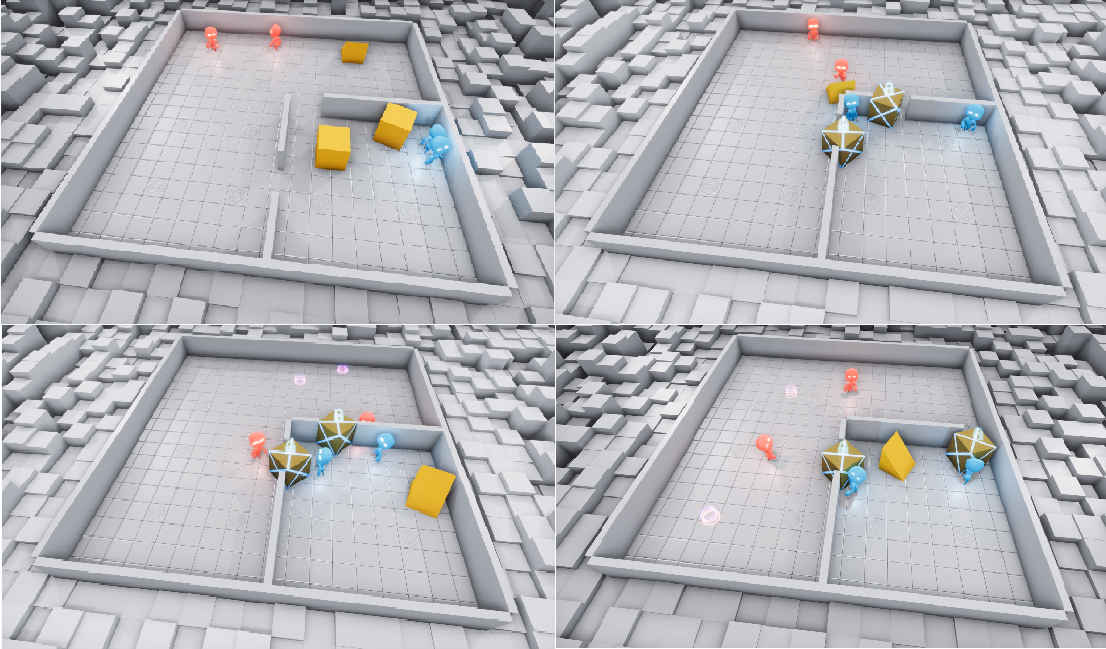
When the same process was repeated for an AI that learned to walk through classical reinforcement learning, it would fall over immediately and would not be able to adapt to the new environment.



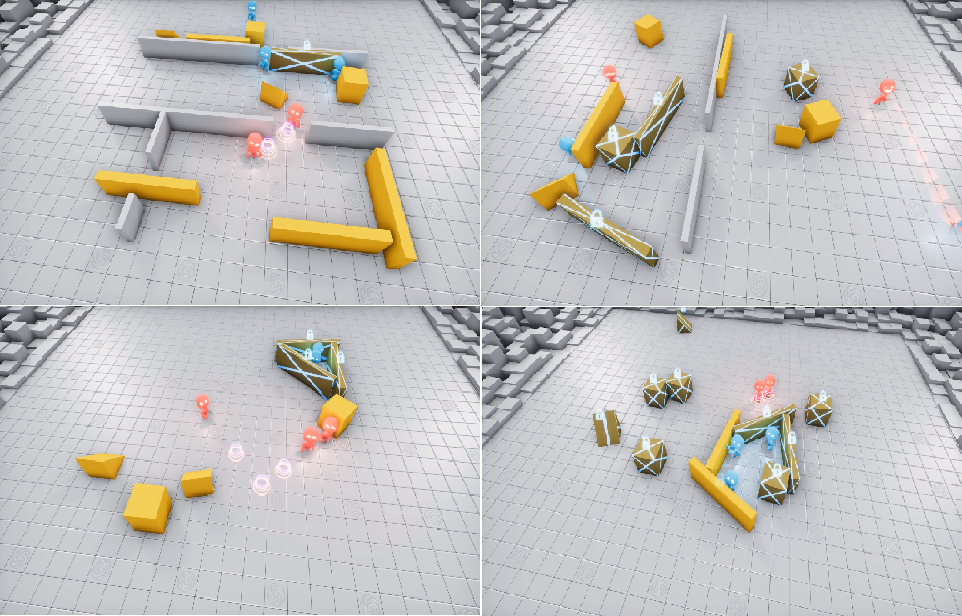
This is an example of AGI vs ANI. The classical methods was able to create an AI that walks and is probably good at walking, but aside from the function of walking it is not able to extrapolate what it learned in a new environment or task. On the other hand the Self-Play method is able to produce an AI that is not only able to do its task but as the same time adapt to new and unseen environments. However OpenAI did state that one flaw of the Self-Play algorithm is the potential of overfitting. Since these AI are learning are learning from their past selves OpenAi realized that sometimes agent would generate strategies that are specifically tailored to counter a specific opponent. Thus when introduced to new opponent with different characteristics it would fail. OpenAI did describe how they dealt with this by pitting the AI against other Self-Play agents that were trained in parallel as well as strategies from previous training processes.

**Multi-Agent Interactions**

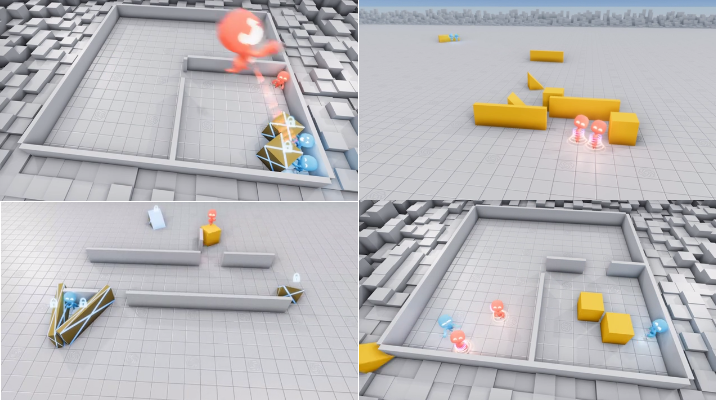
In this last section I would like to talk about how OpenAI was able to implement the Self-Play with multiple Agents in different environments. Just like in real life (IRL) when human interact with each other, the possibilities of what we can do and accomplish is exponentially greater with each addition of a human. The same concept is expressed here through a game of hide and seek. Unlike the previous examples, the environment here is exponentially more complex (multiple agents, moveable blocks, different maps etc.). The goal of the red team (seekers) is to get the blue team (hiders) in their line of sight while the blue team is to hide from the red. The AIs learn to interact with the environment to block certain entrances thus forces the red team to use the ramp to climb over the wall, and as a result prompts blue into hiding the ramp before the round starts (from previous section, agents are generating specific strategies to counter the other team).



The agents produce similar result even in different environments and different amount of agents.

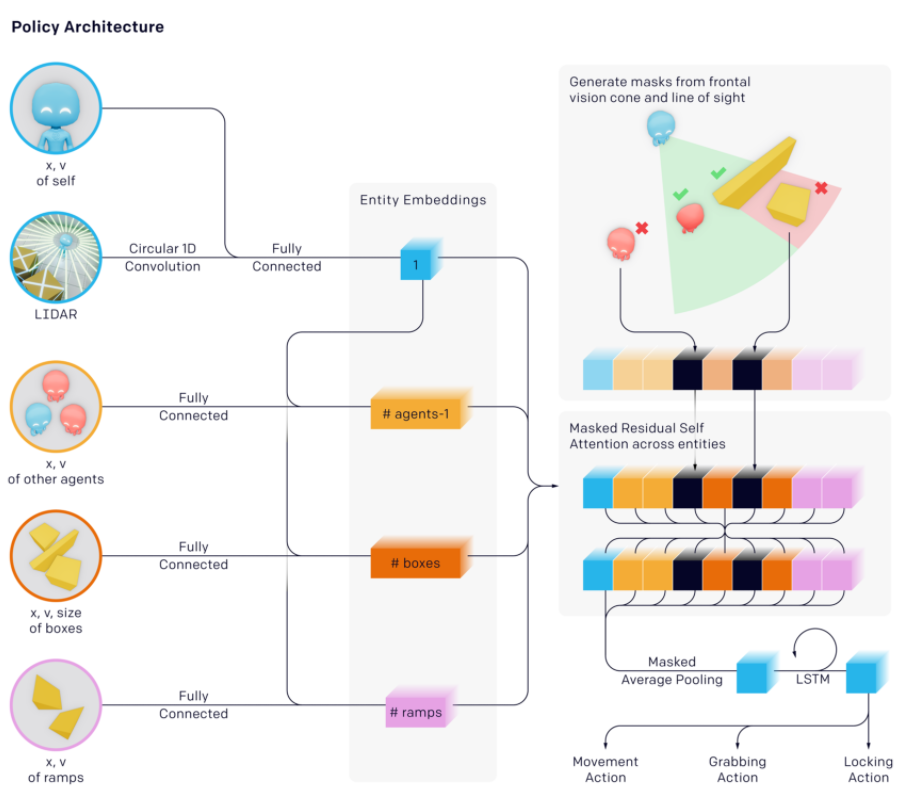


Since this game is built using some physics engine, there are times where the agents found ways to abuse the physics. Such as finding ways the catapult themselves or objects into the air, or surfing a block, or just running infinitely into the distance.

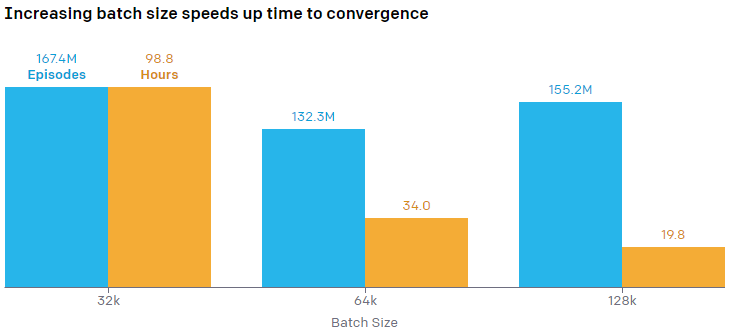


**Training the Agents**

We saw the agents were able to perform in different environments with small permutations of the objects and agents. This is because the agents use an “**entity-centric state-based** representation of the world, which is **permutation invariant**(unaffected by permutations of the objects in the environment) with respect to objects and other agents”. Furthermore the objects in the environment are embedded and masked using a **masked residual self-attention block** (<https://arxiv.org/pdf/1706.03762.pdf>). In other words, objects not in the line of sight of the agents are not used as part of the prediction of the actions. We can see in the chart below, the NN takes into account all objects in the environment (self, LIDAR (field of view), other agents etc.) embeds them before masking them using transformer like methods. Then the embedded features in the agents LIDAR are processed and through their algorithm outputs and action the agent should be taking at the point in time. The masked features are constantly being calculated and thus results in a continuous output of actions.

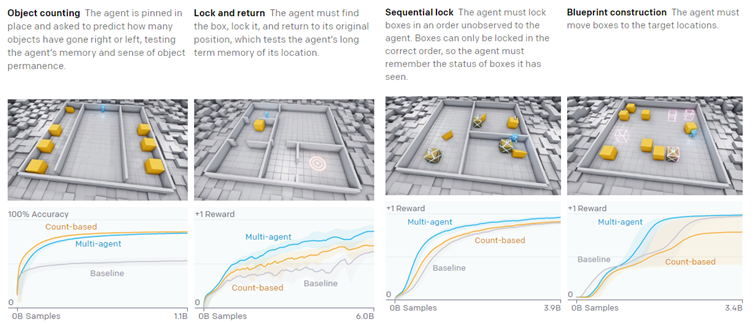


OpenAI also found that larger batch sizes greatly affected the speed at which agents were able to learn. The convergence time shown below measures the amount of time needed for hiders to learn ramp defense (block entrance and hide or lock ramp). Furthermore, they found that batches below 32K never reached this stage.



**Transfer and Evaluation**

Lastly we will talk about how OpenAI evaluated the agents and their learning progress. OpenAI stated that looking at just Reward or metrics like ELO and Trueskill are insufficient in telling us weather the agent is improving in terms of adapting new skills or just improving a previously learned skill. In other words did the agent learn to make fort with block OR just managed to find a more optimal root to walk. This is where they looked at the transferability of Self-Play AI skills in different domain-specific intelligence tests. The results form these specific test can act as a quantitative measure of representation quality or skill.



**Conclusion**

We have come quite far from ANI to AGI. Although AGI is still imperfect we evidence from the gaming community that AGI is slowly developing to require less and less human interaction. The results from Deep Mind’s AlphaGoZero shows that AI which learned by itself is better than its predecessor which learned from human data. Furthermore, OpenAI’s research on Self-Play showed that AIs trained in this manner are more able to adapt to new environments while classical reinforcement methods would fail. Lastly we saw when multiple Self-Play AI interact, they can find new ways to interact with the environment, even in ways that human developers did not foresee. All these are signs that AI is slowly moving away from ANI and towards AGI, and hopefully one day ASI.

**References:**

# Emergent Tool Use from Multi-Agent Interaction

[**https://openai.com/blog/emergent-tool-use/**](https://openai.com/blog/emergent-tool-use/)

[**https://arxiv.org/pdf/1909.07528.pdf**](https://arxiv.org/pdf/1909.07528.pdf)

# Competitive Self-Play

[**https://openai.com/blog/competitive-self-play/**](https://openai.com/blog/competitive-self-play/)

[**https://arxiv.org/pdf/2002.04017.pdf**](https://arxiv.org/pdf/2002.04017.pdf)

## A Study of Count-Based Exploration for Deep Reinforcement Learning <https://openreview.net/forum?id=SyOvg6jxx>

Attention Is All You Need

[**https://arxiv.org/pdf/1706.03762.pdf**](https://arxiv.org/pdf/1706.03762.pdf)