Open AI Five

Steven Kollortz

Heidelberg University

July 25, 2019

Overview

Why choosing DotA

- One of the most popular games on twitch
- Runs on Linux
- Supports an API
- Partially-observed state
- High-dimensional, continuous action and observation space
- Long term planning
- Hoped that in order to solve it, it would require new techniques

Complexity

- An average game lasts around 45 minutes
- With 30 frames per second resulting in 80000 ticks
- OpenAI Five observes every fourth frame

Game	**Length**
Chess	40
Go	150
DotA 2	20000

Table: Number of moves before a game usually ends

- 170,000 possible actions per hero with an average of 1000 valid each tick

Complexity

- Big observation space
- 20000 numbers representing what a human would be able to see as well
- Mostly floating points

Game	Observation space
Chess	8x8 board with 6 pieces plus minor history
Go	19x19 board with 2 pieces plus Ko
DotA 2	20000 numbers

Table: Size of observation space

Timeline

- November 2016 development started
- May 2017 1.5k mmr tester (bottom 15%) is better than the bot
- Early June bot beats 1.5k mmr player
- Late June: bot beats 3k mmr
- July bot beats 7.5k mmr

Timeline

- August 7th beat Blitz (6.2k former pro) 3-0
  Pajkatt (8.5k pro) 2-1 and CC&C (8.9k pro) 3-0.
  They bet Sumail (8.3k pro, top 1v1 player) would win against the bot.
- August 9th beat Arteezy (10k pro, top player) 10-0. He says Sumail could figure out this bot.
- August 10th beat Sumail 6-0, Sumail said it is unbeatable.
- Sumail also played the August 9th version where he goes 2-1
- A lot of people play the bot afterwards, but do not win in a standard game.
- September 7th the first pro beat it with normal gameplay.

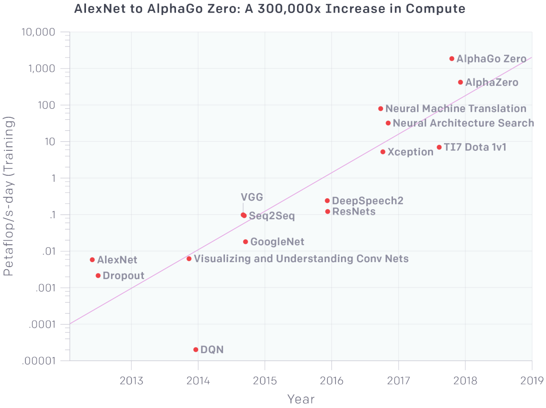
Timeline

- June 2018 OpenAI five on amateur / semi pro level (4-6k mmr) with restricted rules
- Early August 2018 plays on roughly 6-7k mmr with lightly restricted rules (18 heros)
- Loses against against pro teams (7-8k mmr) at The International 8
- Wins against different pro teams 2-0 from October to February 2019, most notably Alliance with team earnings over 3 million dollar.

Timeline

- April 2019 defeats OG, the winner of TI 8.
- April 2019 OpenAI five arena opens where it ends up with a score of 7215–42.
- 10 losses were against the same team.

Computation

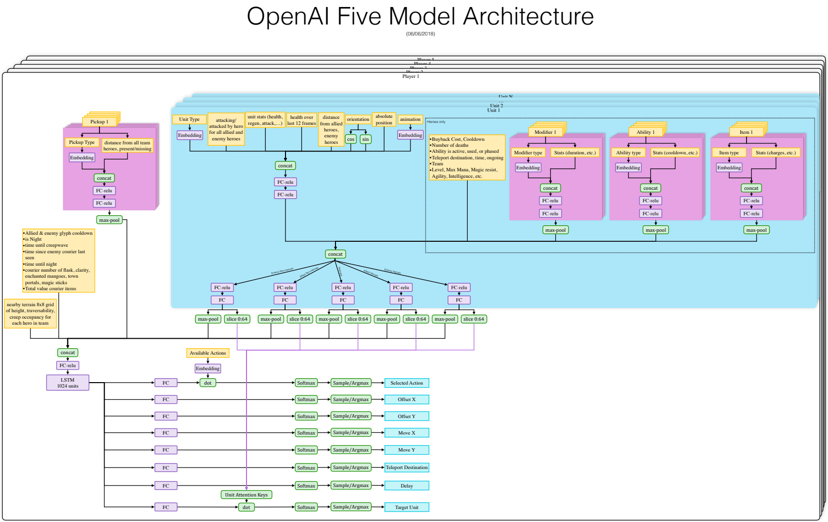


- A doubling of computation every 3.5 months

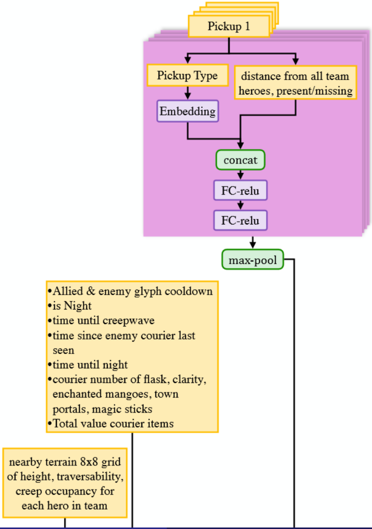
Computation

	1v1 bot	OpenAI five
CPUs	60000 cores on Azure	128000 cores on GCP
GPUs	256 K80	256 P100
Experience	300 years per day	180 years per day per hero
Observation size	3.3 kB	36.8 kB
Observations per second	10	7.5
Batch size	8,388,608	1,048,576
Batches per minute	20	60

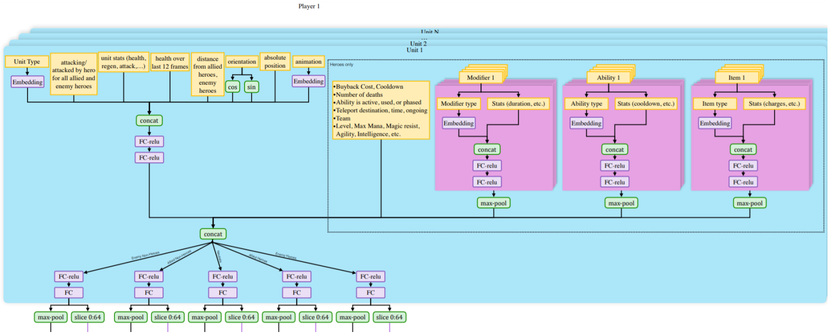
Architecture



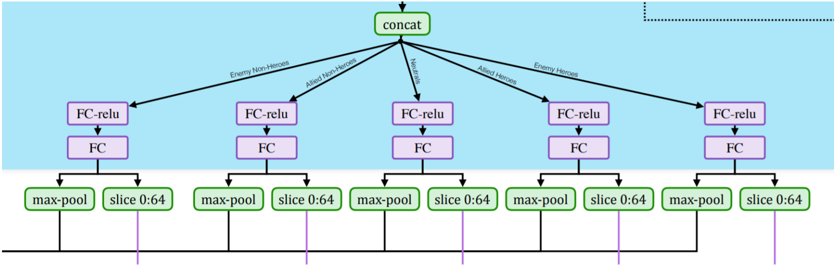
Architecture



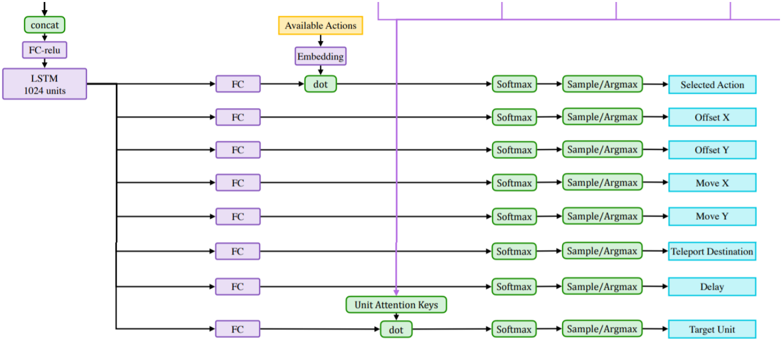
Architecture



Architecture



Architecture



Bots perspective

Proximal Policy Optimization

- Performs comparably to Trust Region Policy Optimization and Actor Critic with Experience Replay
- Easier to implement
- Easier to tune

Proximal Policy Optimization

- TRPO and ACER approximate the second order derivative and its inverse
- PPO uses multiple epochs of stochastic gradient descent to perform each policy update
- Reduce the amount of bad decisions by penalizing or clipping the difference between the old and the new policy
- Clipping yielded best results

Learning

- Inverse reinforcement learning
- Self play
- 80% against itself and 20% against past versions
- Cost intensive
- Transfer learning

Rewardfunction

Score	Weight
Experience	0.002
Gold	0.006
Mana	0.75
Hero Health	2
Last Hit	0.16
Deny	0.2
Kill	-0.6
Death	-1.0
Mega creeps	4.0
Win	2.5

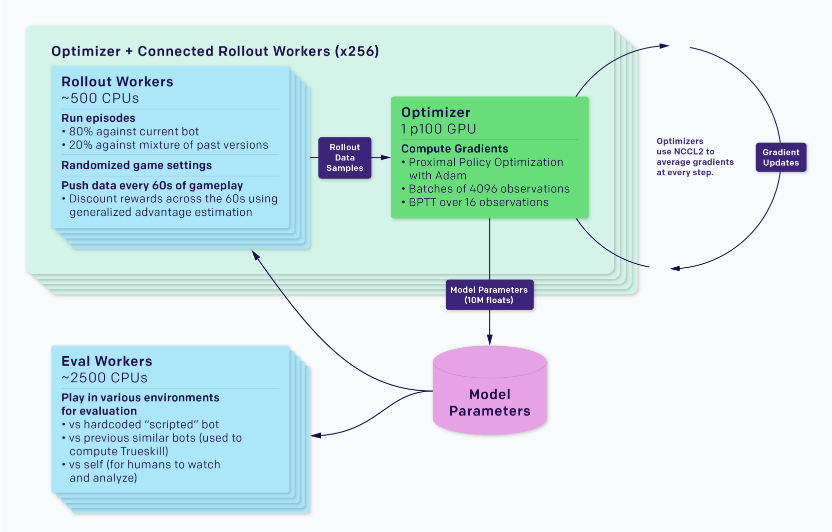
Rewardfunction

- Negative reward for leaving the lane early in training
- Zero sum rewards
- No communication channel
- Teamspirit is controlled by a parameter τ
  $hero\_rewards[i] = \tau * mean(hero\_rewards) + (1 - \tau) * hero\_rewards[i]$
- τ anneals from 0.2 to 0.97 during training
- Later rewards are discounted by half, roughly every 10 minutes

Rapid

- Rapid is a reinforcement learning training system
- Supports Kubernetes, Azure and GCP
- Allows to run PPO in massive scale
- Synchronous gradient descent globally synchronized
- 58MB of parameters have to be synchronized
- takes 0.3 seconds to synchronize 512 GPUs

Rapid


Optimizer + Connected Rollout Workers (x256)
Rollout Workers
~500 CPUs
Run episodes
• 80% against current bot
• 20% against mixture of past versions
Randomized game settings
Push data every 60s of gameplay
• Discount rewards across the 60s using generalized advantage estimation
Rollout Data Samples
Optimizer
1 p100 GPU
Compute Gradients
• Proximal Policy Optimization with Adam
• Batches of 4096 observations
• BPTT over 16 observations
Optimizers use NCCL2 to average gradients at every step.
Gradient Updates
Model Parameters (10M floats)
Model Parameters
Eval Workers
~2500 CPUs
Play in various environments for evaluation
• vs hardcoded "scripted" bot
• vs previous similar bots (used to compute Trueskill)
• vs self (for humans to watch and analyze)

References

https://openai.com/blog/ai-and-compute/
https://liquipedia.net/dota2/The_International/2018/Main_Event
https://openai.com/blog/how-to-train-your-openai-five/
https://arena.openai.com/#/results
https://openai.com/blog/openai-five/
https://openai.com/blog/more-on-dota-2/
https://openai.com/five/ https://d4mucfpksywv.cloudfront.net/research-covers/openai-five/network-architecture.pdf
https://openai.com/blog/openai-baselines-ppo/#ppo
https://medium.com/@jonathan_hui/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12
https://gist.github.com/dfarhi/66ec9d760ae0c49a5c492c9fae93984a
https://arxiv.org/pdf/1611.01224.pdf
https://arxiv.org/pdf/1707.06347.pdf

Questions?