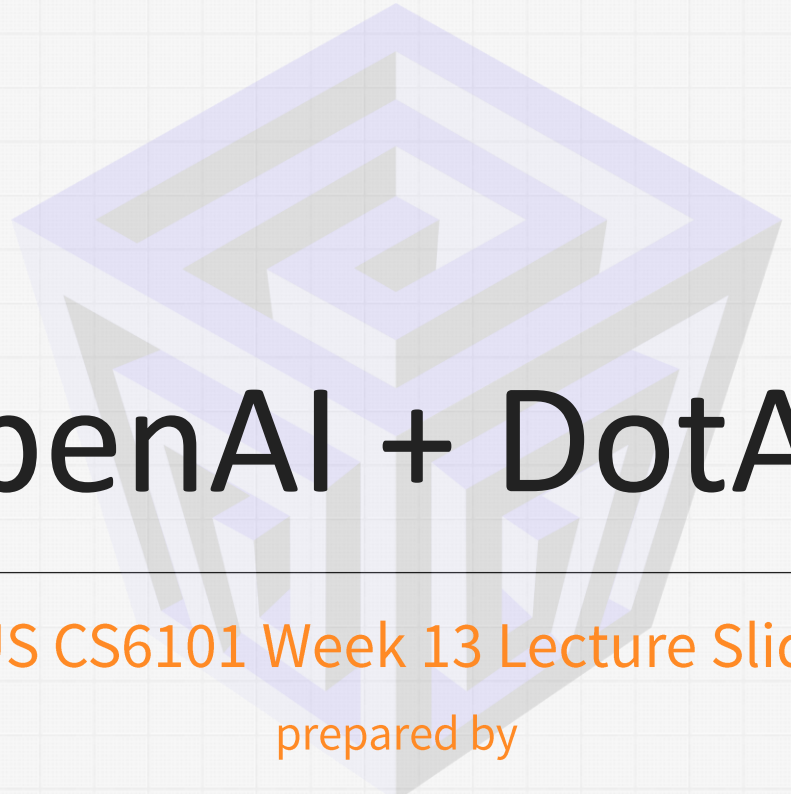


# OpenAI + DotA 2

NUS CS6101 Week 13 Lecture Slides

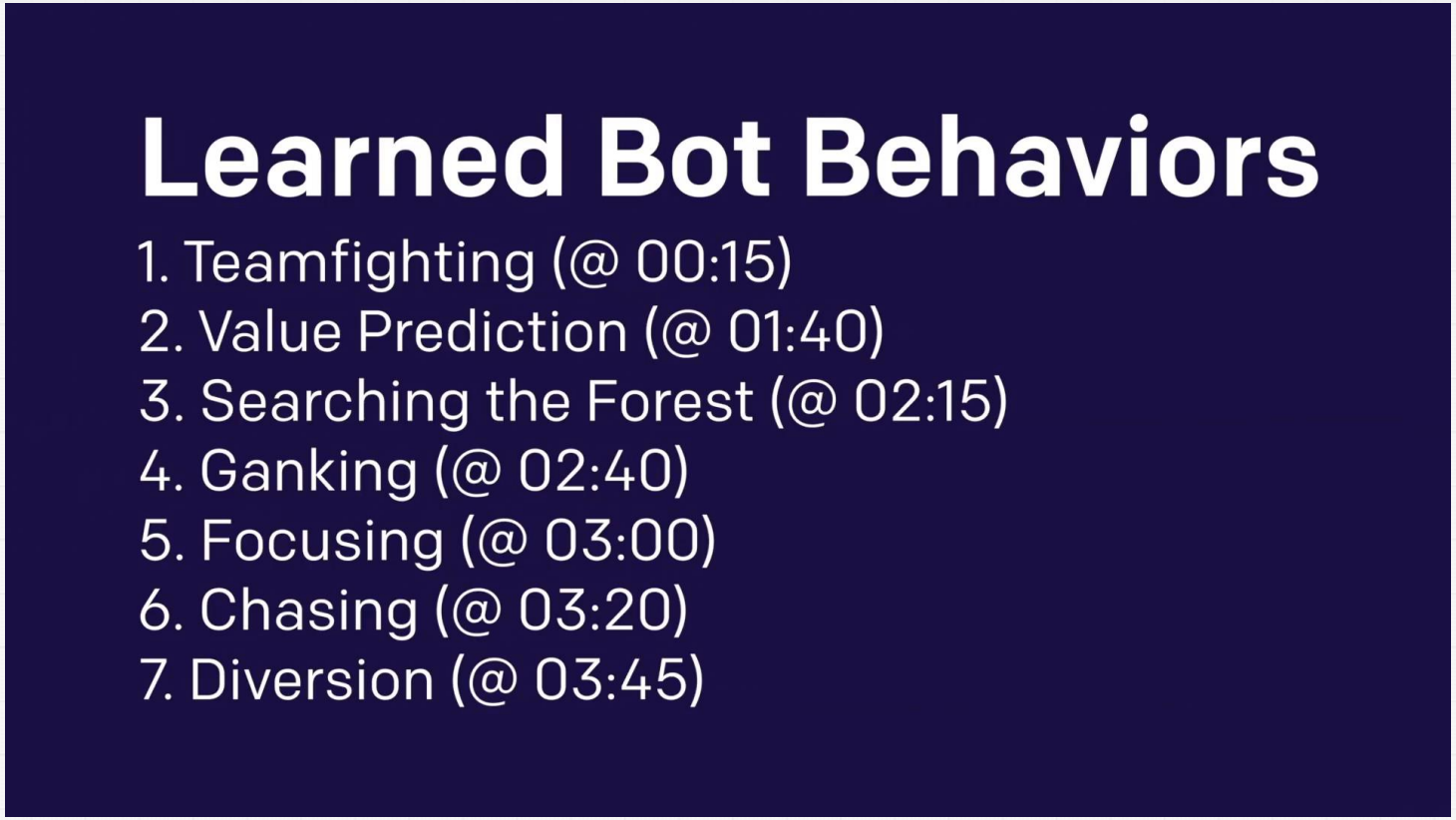
prepared by

Markus Kirchberg

# DotA (Defense of the Ancients) 2 Gameplay



- Multiplayer Online Battle Arena game.
- Steep learning curve.
- Map based; pits 5 players on ‘Dire’ vs. 5 players on ‘Radiant’. Each player controls one hero.
- 115+ heroes: unique stats, spells & tree talents.
- Objective: Destroy all the enemy towers and ultimately their throne.
- Gameplay: Fight the enemy team, destroy enemy buildings, try to die as little as possible, level up, buy items, and kill enemy players when possible.
- Average game time: 45-50 minutes.


OpenAI Five: Dota Gameplay; Source: https://www.youtube.com/watch?v=UZHTNBMAfAA

# DotA and StarCraft II Challenges [1, 2, 3]

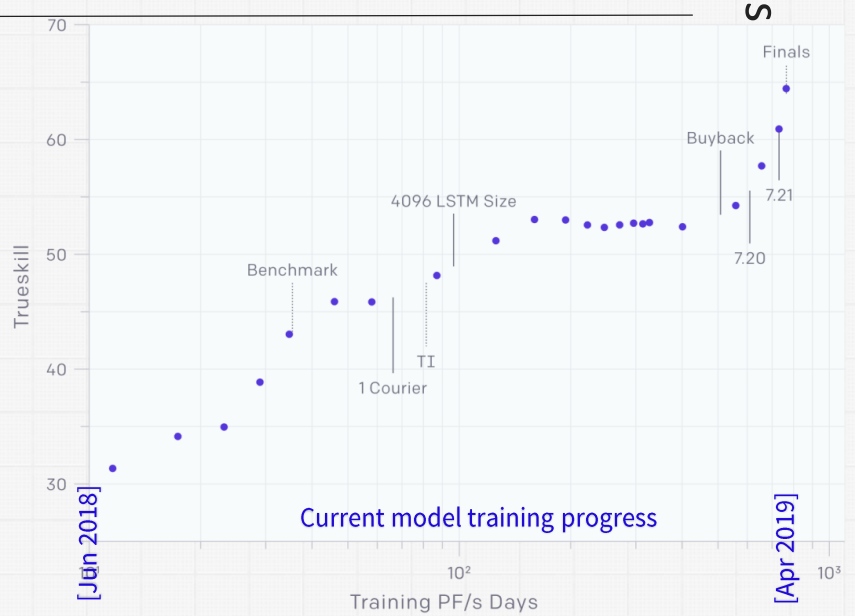


| | Atari Games | Go | DOTA 2 | StarCraft II |
|---|---|---|---|---|
| **Information Type** | **Various** | Perfect | **Imperfect** | **Imperfect** |
| **Players** | Single Player | **Multi-player** | **Multi-player** | **Multi-player** |
| **Action Space** | **Continuous & Discrete** | Discrete | **Continuous & Discrete** | **Continuous & Discrete** |
| **Possible Actions** | 17 | 361 | **Millions** | **Millions** |
| **No. of Moves per Game** | 100's of moves | 100's of moves | **1,000's of moves** | **1,000's of moves** |

- **Long time horizons** (30 fps @ 45min → 80,000 ticks per game; OpenAI Five observes every fourth frame → 20,000 moves).
- **Partially-observed state** (units and buildings can only see the area around them).
- **High-dimensional, continuous action space** (space discretized into 170,000 possible actions per hero with an average of ~1,000 valid actions each tick).
- **High-dimensional, continuous observation space** (a large continuous map containing ten heroes, dozens of buildings, dozens of NPC units, and a long tail of game features such as runes, trees, and wards).
- **Complex rules** and **constantly changing the environment semantics** (game updates released every other week).

# OpenAI DotA Progress Summary [1]






- **Nov 2016**: Beginning of development on the algorithms used for Dota2.
  - Chosen for its popularity, native Linux support and ready application programming interface (API).
- **Mar 2017**: First agent defeated DotA (training) bots; but got confused against humans.
- **Aug 2017**: **First public demonstration** at The International 2017 (DotA annual tournament).
  - **OpenAI 1v1 defeated Dendi, a professional DotA player** (in a live one-on-one matchup).
  - Launched OpenAI Five initiative.
- **Apr 2018**: OpenAI Five beats internal, scripted baseline.
- **May 2018**: OpenAI Five draws (1-1) against amateur OpenAI employee team (2.5k MMR; 46th %ile).
- **Jun 2018**: OpenAI Five decisively won all its games versus amateurs & semi-professionals playing as a team (2.5-4k MMR; 46th-90th %ile).
- **Jun 2018**: OpenAI Five won two of its first three games versus amateur team (4.2k MMR; 93rd %ile) and semi-pro team (5.5k MMR; 99th %ile).
- **Aug 2018** @ The International (TI) 2018: **OpenAI Five lost two games against professional teams**.
- **Apr 2019**: **Public demonstration at which OpenAI Five won two back-to-back games versus The International 2018 champions**.
  - Public demo of cooperative play (OpenAI Five bots play alongside human players).
- **Apr 2019**: [upcoming] From Apr 18th–21st, OpenAI Five plays the Internet (https://arena.openai.com/).

# OpenAI DotA Approach [1]



## Generic Approach

- System learns using a **massively-scaled version of Proximal Policy Optimization (PPO)**.
- Both OpenAI 1v1 and OpenAI Five learn 'entirely' from self-play.
- Focus: **Running today's RL algorithms at sufficient scale** and with a reasonable way of exploration.
- **Training starts with random parameters** and do not use search or bootstrap from human replays.
- Agent trained to **maximize the exponentially decayed sum of future rewards**, weighted by an exponential decay factor called γ.

## Weakness (Aug 2018 Defeat)

- Median score for last-hitting (ie, delivering the killing blow; hero dealing the killing blow will be awarded a bounty).

| | OPENAI 1V1 BOT [mid-2017] | OPENAI FIVE TI version [Jun 2018] | [Apr 2019] |
|---|---|---|---|
| **CPUs** | 60,000 CPU cores on Azure | 128,000 preemptible CPU cores on GCP | **8x more training compute** |
| **GPUs** | 256 K80 GPUs on Azure | 256 P100 GPUs on GCP | |
| **Experience collected** | ~300 years per day | ~180 years per day (~900 years per day counting each hero separately) | **~250 years per day** (4.5 times more experience years than TI version) |
| **Size of observation** | ~3.3 kB | ~36.8 kB | |
| **Observations per second of gameplay** | 10 | 7.5 | |
| **Batch size** | 8,388,608 observations | 1,048,576 observations | |
| **Batches per minute** | ~20 | ~60 | |

# OpenAI DotA ‘Cheats’ [1, 3]



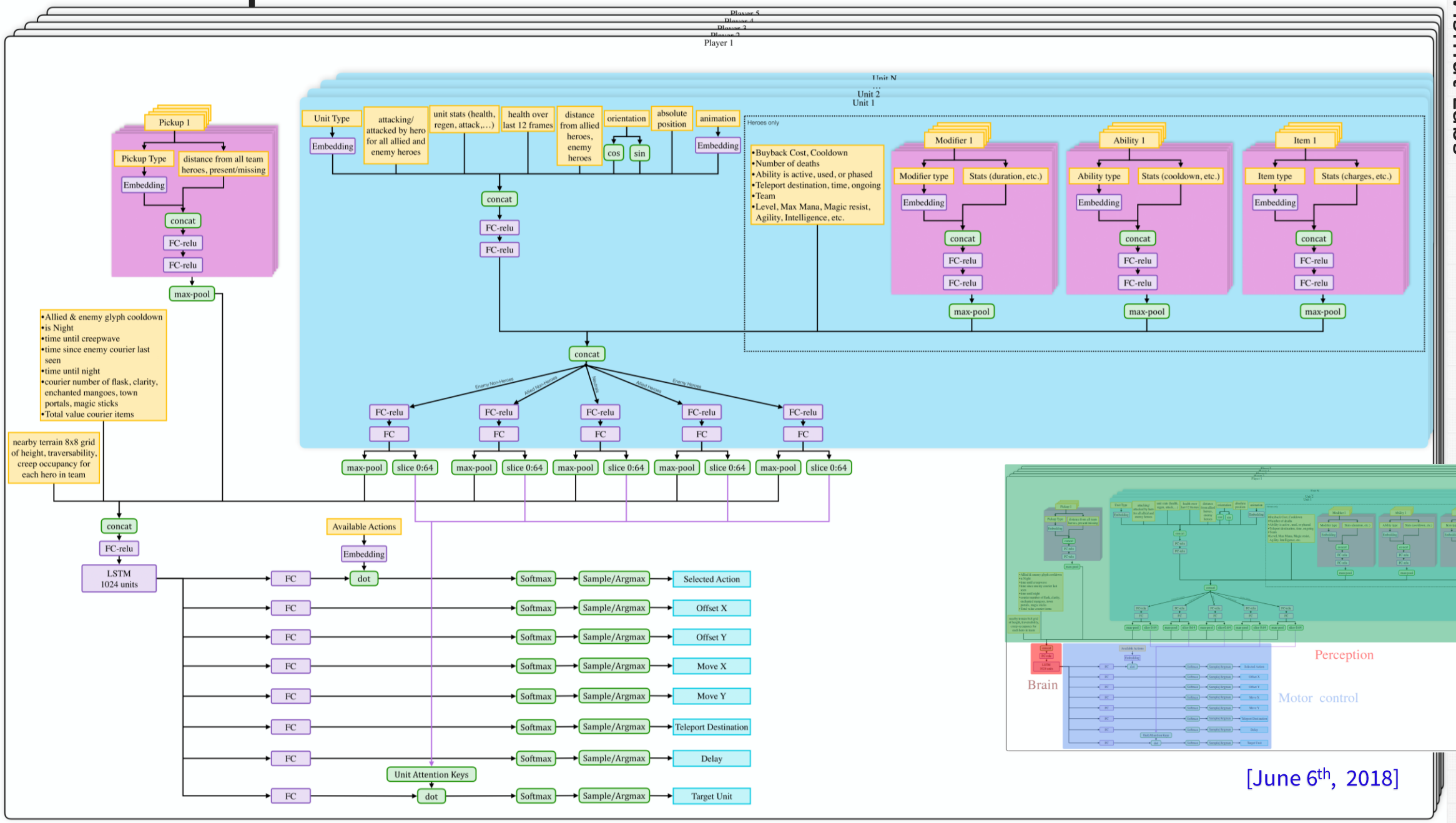
- **API and superhuman speed**: The bots were trained using API to get precise game state as thousands of exact numbers.
  - Allows the bots to have ‘superhuman’ accuracy and speed → very hard to beat in head-to-head short term fights that don’t require much long-term strategy.
  - OpenAI: “*Our method isn’t fundamentally tied to observing state, but just rendering pixels from the game would require thousands of GPUs.*”
- **Simplifying some tasks during training**: Agents encouraged to explore important but possibly challenging strategic decisions.
- **Hand-designing the model**: Designed a complex neural net architecture specifically for the task.
- **Limitations on the game**: Multiple restrictions were placed on standard gameplay.
  - E.g., map visibility and reduced roster of playable heroes.
- **Reward shaping** → learning is not purely from self-play!
  - Heavy use of reward tuning (ie, short term goals are explicitly rewarded instead of only focusing on winning the game).
  - Bots are guided to learn certain behaviours.

**Human View**

**AI View**

| | | | | | |
|---|---|---|---|---|---|
| 3.006 | -1.386 | -0.4695 | 0.883 | 1 | 0.84 |
| -0.3154 | -0.5425 | -0.5 | 0.866 | 0 | 0.82 |
| 3.11 | -1.36 | -0.9336 | 0.3584 | 1 | 0.78 |
| -2.324 | 2.863 | 0.9746 | 0.225 | 0 | 0.86 |
| 3.037 | -1.361 | -0.7773 | 0.6294 | 1 | 0.82 |
| -1.387 | 2.951 | 0.988 | 0.1565 | 0 | 0.74 |
| 3.023 | -0.9395 | 0.05234 | -0.9985 | 0 | 0.66 |
| 2.951 | -0.5747 | 0.01746 | 1 | 0 | 0.72 |
| 2.963 | -1.303 | 0.3906 | 0.9204 | 0 | 0.68 |
| 2.834 | -3.164 | 0.01746 | -1 | 0 | 0.68 |
| 3.127 | -1.368 | 0.6562 | 0.755 | 1 | 0.55 |
| 3.088 | -1.366 | 0.4695 | 0.883 | 0 | 0.55 |
| 2.984 | -1.398 | -0.225 | 0.9746 | 1 | 0.55 |
| 3.037 | -1.391 | 0.788 | 0.6157 | 0 | 0.55 |
| 3.076 | -1.438 | 0.883 | 0.4695 | 0 | 0.55 |
| -2.412 | 2.846 | 0.996 | 0.08716 | 1 | 0.3 |

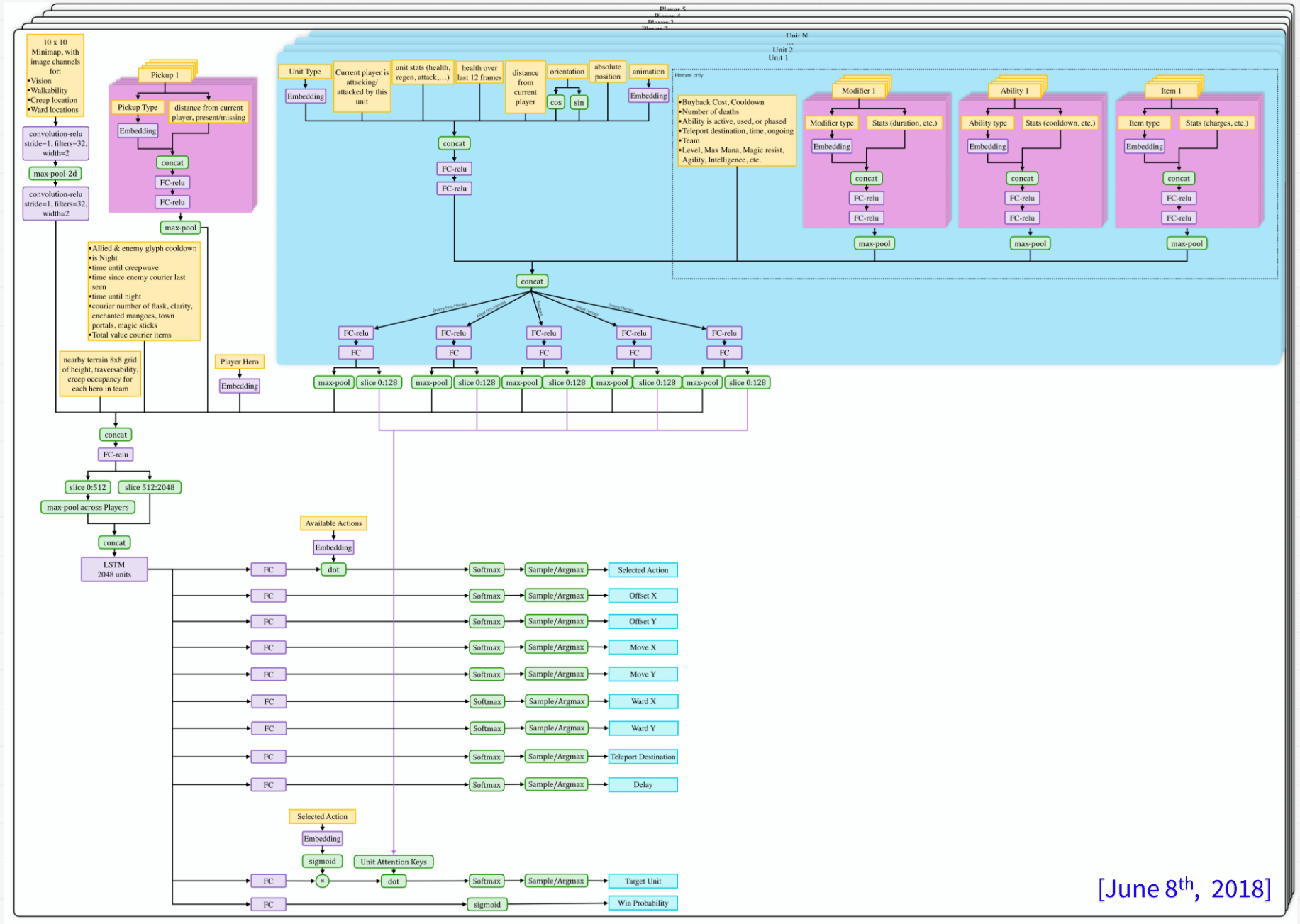
# OpenAI Five Network Architecture [1, 6]




Wismut Labs

[June 6th, 2018]

# OpenAI Five Network Architecture [1, 6]




[June 8th, 2018]

Wismut Labs

# OpenAI Five Model Structure [1]



- Each OpenAI Five network contain a **single-layer, ~~1024 unit~~ ~~2048 unit~~ 4096-units LSTM**:
  - Sees the current game state (extracted from Valve's Bot API).
  - Emits actions through several possible action heads.
  - Each head has semantic meaning (e.g., no. of ticks to delay this action, which action to select, the X or Y coordinate of this action in a grid around the unit, …).
  - Action heads are ***computed independently***.
- Interactive demonstration of the observation space and action space used by OpenAI Five: https://openai.com/blog/openai-five/#modelstructure
  - Views the world as a list of 20,000 numbers.
  - Takes an action by emitting a list of 8 enumeration values.
- OpenAI Five **can react to missing pieces of state** that correlate with what it does see.
  - E.g., shrapnel zones (areas where projectiles rain down on enemies) were not modelled. Yet, OpenAI Five learnt to walk out of (though not avoid entering) active shrapnel zones, as it could see its health decreasing.

# OpenAI Five Exploration [1]



- **Combinatorially-vast space to explore!**
  - Even with restrictions: Hundreds of items, dozens of buildings, spells, and unit types, and a long tail of game mechanics.
- OpenAI Five **learns from** (guided) **self-play** (starting from random weights) → natural curriculum for exploring the environment.
  - To avoid "strategy collapse", the agent trains 80% of its games against itself and the other 20% against its past selves.
  - "*In the first games, the heroes walk aimlessly around the map. After several hours of training, concepts such as laning, farming, or fighting over mid emerge. After several days, they consistently adopt basic human strategies: attempt to steal Bounty runes from their opponents, walk to their tier one towers to farm, and rotate heroes around the map to gain lane advantage. And with further training, they become proficient at high-level strategies like 5-hero push.*"
- To **force exploration in strategy space**, during training (and only during training) → randomized units' properties.

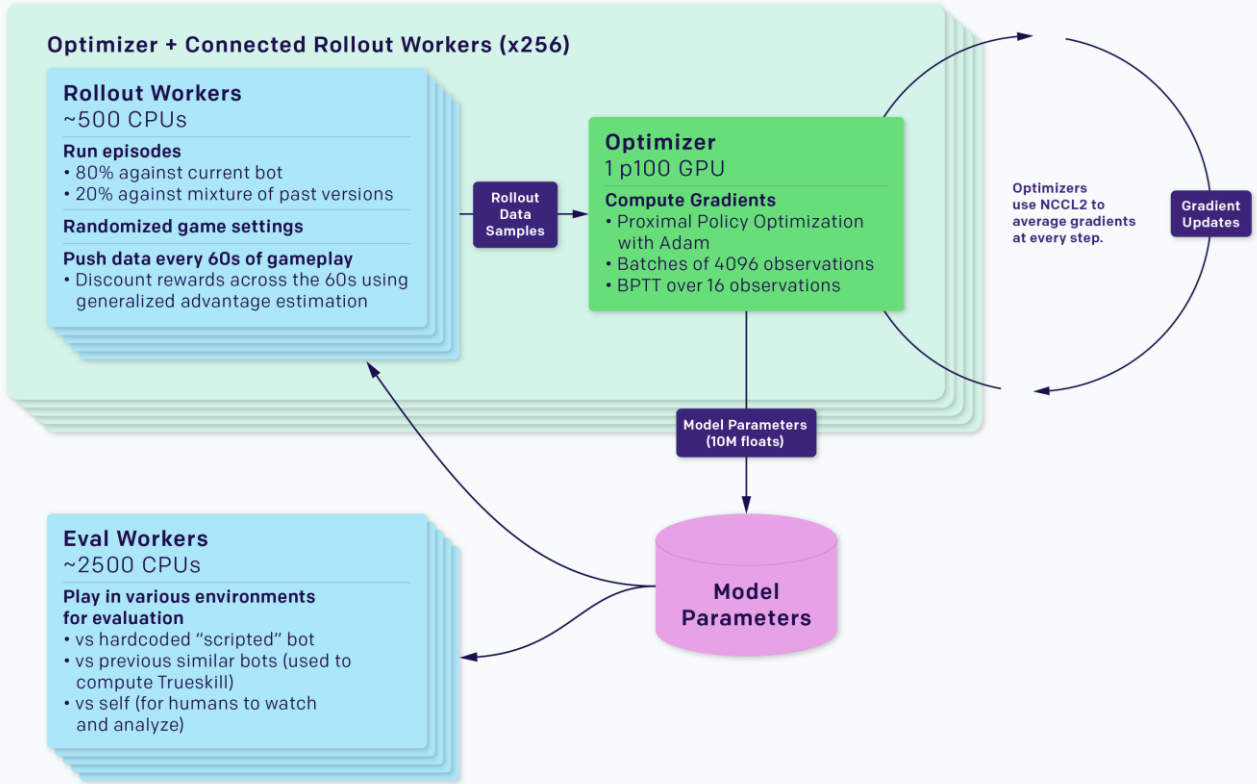
  → OpenAI Five began beating humans.
- Exploration is also helped by a **good reward**.
  - Reward = Metrics humans track: net worth, kills, deaths, assists, last hits, and the like.
  - Each agent's reward is post-processed by subtracting the other team's average reward to prevent the agents from finding positive-sum situations.

# OpenAI Rapid [1] … a general-purpose RL training system



- OpenAI Five is implemented using Rapid, which can be applied to any Gym environment.
- **OpenAI Five training system:**
  - Rollout workers, which run a copy of the game and an agent gathering experience.
    - Sync their experience through Redis to the optimizers.
  - Optimizer nodes, which perform synchronous gradient descent across a fleet of GPUs.
    - Each GPU computes a gradient on its part of the batch, and then the gradients are globally averaged.
  - Evaluation workers, which evaluate the trained agent versus reference agents.
  - Monitoring software (TensorBoard, Sentry & Grafana).
- The latencies for synchronizing model parameters is low enough to be masked by GPU computation, which runs in parallel with it.

→ **Runs PPO at previously unprecedented scale.**



# Proximal Policy Optimization [1, 4, 5]

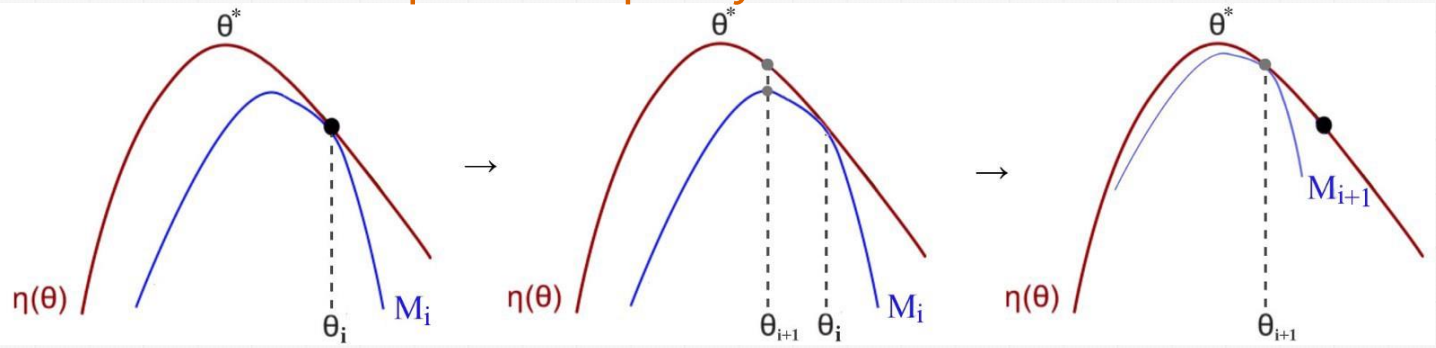


- OpenAI: "*Proximal Policy Optimization (PPO), which perform comparably or better than state-of-the-art approaches while being much simpler to implement and tune.*"

- Policy Gradient methods have convergence problem (→ natural policy gradient).
  - In practice: Natural policy gradient involves a second-order derivative matrix → not scalable.
  - The computational complexity is too high for real tasks.
  - Intensive research to reduce the complexity by approximate the second-order method.

How can we optimize a policy to maximize the rewards?

**Minorize-Maximization (MM) algorithm:**




→ **Find a lower bound M that is easier to optimize.**

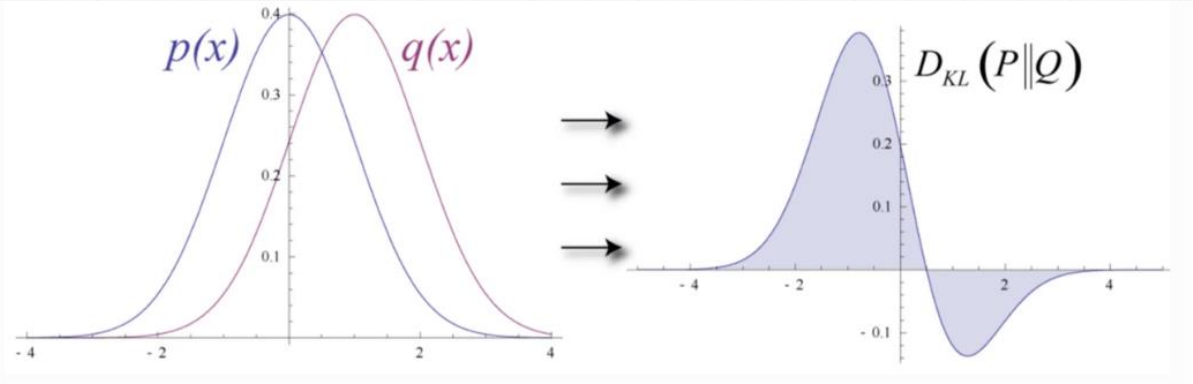
# Proximal Policy Optimization [1, 4, 5]

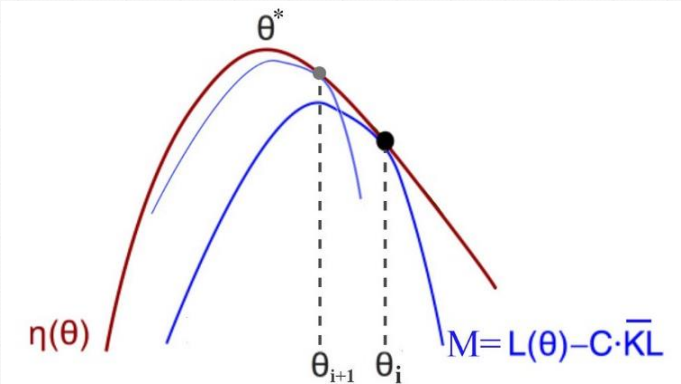


- Two major **optimization methods**: **Line search** (e.g., the gradient descent) vs. **Trust region**.
  - Gradient descent is easy, fast and simple in optimizing an objective function.

- **PPO**: Limit how far we can change a policy in each iteration through the KL-divergence.
  - **KL-divergence** measures the difference between two data distributions p and q.
  - Repurpose: Measure the difference between the two policies → any new policy shouldn't be too different from the current one.

$$D_{KL}(P||Q) = \mathbb{E}_x \log \frac{P(x)}{Q(x)}$$



# Proximal Policy Optimization [1, 4, 5]



- **Intuition:**
  - **L approximates the advantage function locally at the current policy.**
  - It gets less accurate as it moves away from the old policy.
  - This **inaccuracy has an upper bound** → That is the second term in **M**.
  - After considering the upper bound of this error, we can guarantee that the calculated optimal policy within the trust region is always better than the old policy.
  - If the policy is outside the trust region, even the calculated value may be better but the accuracy can be too off and cannot be trusted → abandon.

**Objective**

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right]$$
$$\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta.$$

or

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] - \beta\hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$$

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^T F^{-1} g}} F^{-1} g$$

natural policy gradient

computationally very expensive!

$$F = \nabla^2 f = \begin{pmatrix} \frac{\partial^2 f_1}{\partial x_1^2} & \frac{\partial^2 f_1}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f_1}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f_2}{\partial x_1^2} & \frac{\partial^2 f_2}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f_2}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f_m}{\partial x_1^2} & \frac{\partial^2 f_m}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f_m}{\partial x_1 \partial x_n} \end{pmatrix}$$

# Proximal Policy Optimization [1, 4, 5]



- **Two approaches to address this problem**:
  1. Approximate some calculations involving the second order derivative and its inverse to lower the computational complexity; or
     - TRPO & ACKTR follow this approach.
  2. Make the first order derivative solution (e.g., gradient descent) closer to the second-order derivative solution by **adding soft constraints**.
     - PPO follows this approach.

- PPO: Instead of imposing a hard constraint, it **formalizes the constraint as a penalty in the objective function**.
  - Use a first-order optimizer like the Gradient Descent method to optimize the objective.
  - Even we may violate the constraint once a while, the damage is far less and the computation is much simple.
  - Offers a better insurance that we are optimizing within a trust region → The chance of a bad decision is smaller.

# Proximal Policy Optimization [1, 4, 5]



- **PPO with Adaptive KL Penalty:**
  - Re-formulate objective → **Change the constraint to a penalty in the objective function**.
  - Penalizes the objective if the new policy is different from the old policy.

**Algorithm 4** PPO with Adaptive KL Penalty

Input: initial policy parameters $\theta_0$, initial KL penalty $\beta_0$, target KL-divergence $\delta$
**for** $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta||\theta_k)$$

  by taking $K$ steps of minibatch SGD (via Adam)
  **if** $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  **else if** $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \leq \delta/1.5$ **then**
    $\beta_{k+1} = \beta_k/2$
  **end if**
**end for**

# Proximal Policy Optimization [1, 4, 5]



- **PPO with clipped objective** (better performance):
  - Maintain **two policy networks** (the policy to refine, and the policy last used to collect samples).
  - **Importance sampling**: Evaluate a new policy with samples collected from an older policy.

**Algorithm 5** PPO with Clipped Objective

Input: initial policy parameters $\theta_0$, clipping threshold $\epsilon$
**for** $k = 0, 1, 2, \ldots$ **do**
Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

by taking $K$ steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k} \left[ \sum_{t=0}^{T} \left[ \min(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k}) \right] \right]$$

**end for**

# Proximal Policy Optimization [1, 4, 5]



- OpenAI:

*“Q-learning (with function approximation) fails on many simple problems and is poorly understood,*

*vanilla policy gradient methods have poor data efficiency and robustness; and*

*trust region policy optimization (TRPO) is relatively complicated, and is not compatible with architectures that include noise (such as dropout) or parameter sharing (between the policy and value function, or with auxiliary tasks).”*

**→ Simplicity Rules**

# Transfer Learning for RL [1]



- The current (mid-April 2019) version of OpenAI Five has been training continuously since June 2018!
  - Despite changes to the model size and the game rules.
  - **For each change, the model was transferred over and training continued** — something that is an open challenge for RL in other domains.

→First time an RL agent has been trained using such a long-lived training run.

# Open Challenges & Moving Forward



- OpenAI:

  *“We’ve seen rapid progress in the past two years on RL capabilities, and we think that Dota 2 will continue to help us push forward what’s possible — whether with achieving competent performance from less data or true human-AI cooperation.”*

- **R&D Challenges** (some of many):
  - Any explicit (or implicit) bridge between short-term and long-term planning.
  - Abstract reasoning within models.
  - More intuitive specification of objectives.

# Reference Materials




1. OpenAI Blog. Available online at: https://openai.com/blog/
2. DeepMind; “AlphaStar: Mastering the Real-Time Strategy Game StarCraft II”. Available online at: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/
3. Thomas C. H. Lux; “How have DOTA and StarCraft wins advanced AI research?”; Skynet Today; 12/Apr/2019. Available online at: https://www.skynettoday.com/editorials/openai-dm
4. John Schulman et.el.; “Proximal Policy Optimization Algorithms”; 2017. Available online at: http://arxiv.org/abs/1707.06347
5. Jonathan Hui; “RL — Proximal Policy Optimization (PPO) Explained”; Sep 17, 2018. Available online at: https://medium.com/@jonathan_hui/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12
6. Tambet Matiisen; “The use of embeddings in OpenAI Five”; Sep 9, 2018. Available online at: https://neuro.cs.ut.ee/the-use-of-embeddings-in-openai-five/