# Grandmaster level in StarCraft II using multi-agent reinforcement learning

(Oriol Vinyals et al, 2019.10.30)

2020. 09. 14
Donggu Kang

# 목차

- How hard?
- How to do?
- Architecture
- SL
- RL
- Exploration
- League training
- Infra structure

# How hard?

- No single best strategy. : 가위바위보처럼 베스트 전략은 없음, 게임이론
- Imperfect information : 전장의 안개, 미니맵으로만 알수 있는 정보 제한
- Delayed reward : action의 영향이 바로 나타나지 않음
- Real time : 생각할 시간이 없음. 판단했으면 바로 해야함.
- Large action space : 수많은 유닛,건물,행동의 종류가 서로 맞물려서 영향을 끼침 대략 매 step마다 10^26

# How to do?

- Imitation learning + SL + RL
- League training : PFSP(Prioritized fictitious self-play)
- So many method…
- 사람과 비슷하게 하기 위해 APM limit, Delay 적용

etc. scatter connections, transformer, pointer network, LSTM, Attention, ResNet, V-trace, TD(lambda) UPGO, KL divergence, L2 reg, Adam, pseudo-reward, replay buffer

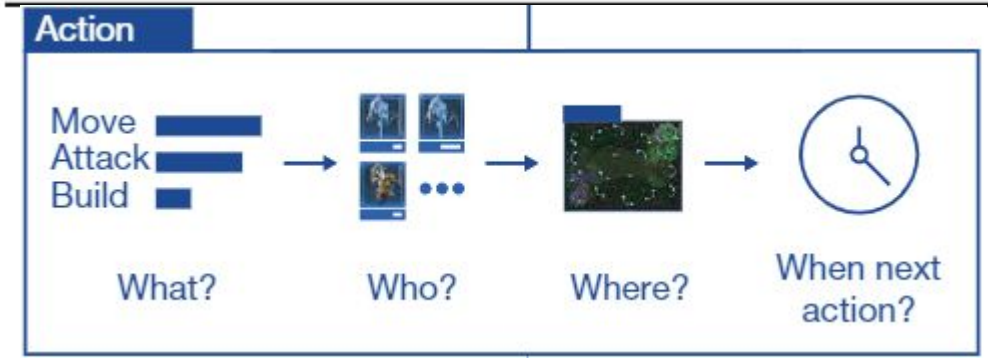
# Architecture

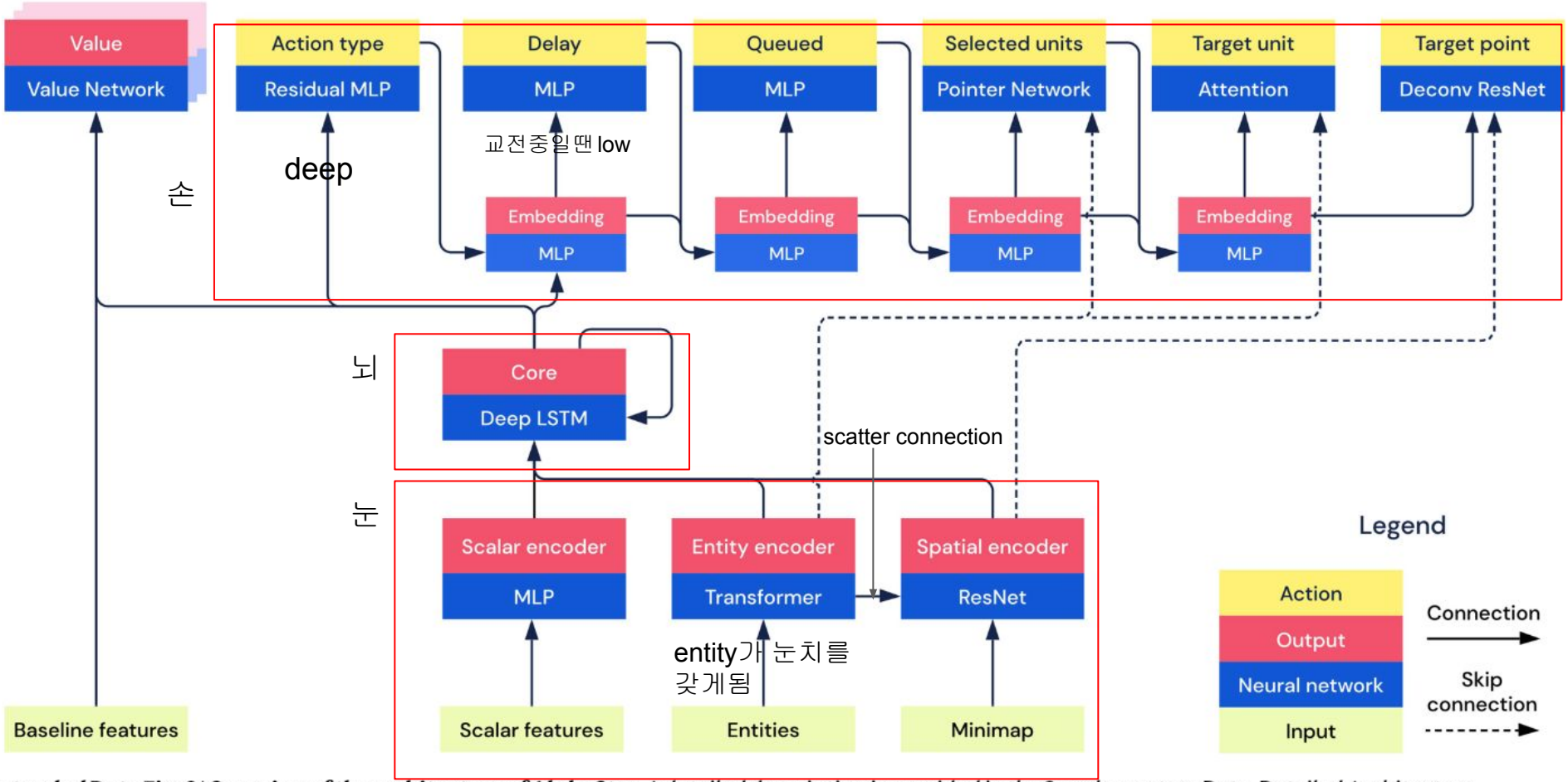
- Policy : $\pi_\theta(a_t \mid s_t, z)$
- S : $s_t = o_{1:t}, a_{1:t-1}$
- Z : strategy statistics : 현재 step에서 a 확률분포에 영향
- O : vector encoded by LSTM(대충 step마다 기억을 저장하는 의미)
- A : sampled auto-regressively : output이 다음 input으로 들어감

대충 139 million weights 중 55 mil개만 inference에 쓰임.

| Category | Field | Description |
|---|---|---|
| Entities: up to 512 | Unit type | E.g. Drone or Forcefield |
| | Owner | Agent, opponent, or neutral |
| | Status | Current health, shields, energy |
| | Display type | E.g. Snapshot, for opponent buildings in the fog of war |
| | Position | Entity position |
| | Number of workers | For resource collecting base buildings |
| | Cooldowns | Attack cooldown |
| | Attributes | Invisible, powered, hallucination, active, in cargo, and/or on the screen |
| | Unit attributes | E.g. Biological or Armored |
| | Cargo status | Current and maximum amount of cargo space |
| | Building status | Build progress, build queue, and add-on type |
| | Resource status | Remaining resource contents |
| | Order status | Order queue and order progress |
| | Buff status | Buffs and buff durations |
| Map: 128x128 grid | Height | Heights of map locations |
| | Visibility | Whether map locations are currently visible |
| | Creep | Whether there is creep at a specific location |
| | Entity owners | Which player owns entities |
| | Alerts | Whether units are under attack |
| | Pathable | Which areas can be navigated over |
| | Buildable | Which areas can be built on |
| Player data | Race | Agent and opponent requested race, and agent actual race |
| | Upgrades | Agent upgrades and opponent upgrades, if they would be known to humans |
| | Agent statistics | Agent current resources, supply, army supply, worker supply, maximum supply, number of idle workers, number of Warp Gates, and number of Larva |
| Game statistics | Camera | Current camera position. The camera is a 32x20 game-unit sized rectangle |
| | Time | Current time in game |

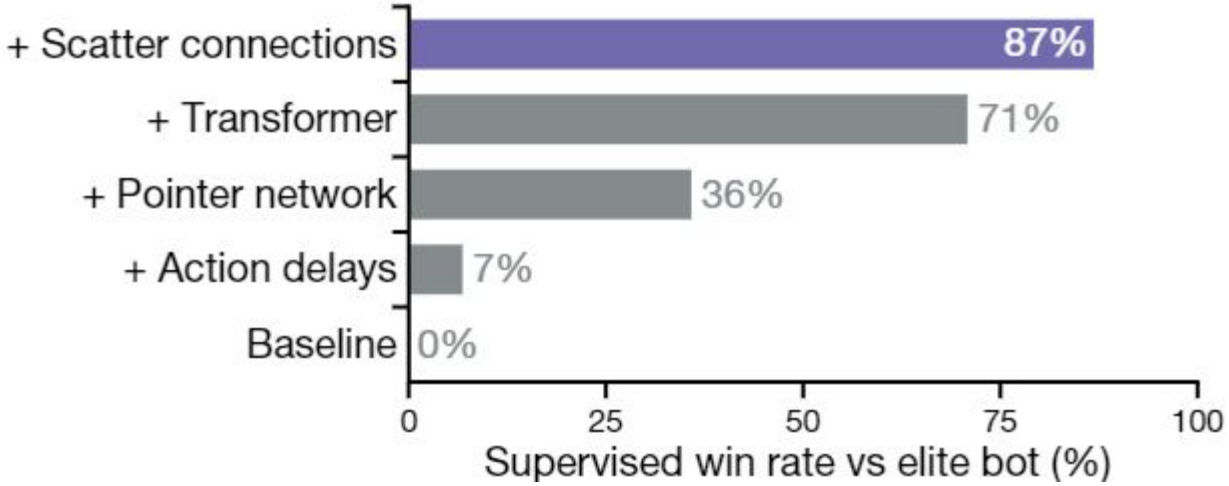
| Field | Description |
|---|---|
| Action type | Which action to execute. Some examples of actions are moving a unit, training a unit from a building, moving the camera, or no-op. See PySC2 for a full list[7] |
| Selected units | Entities that will execute the action |
| Target | An entity or location in the map discretised to 256x256 targeted by the action |
| Queued | Whether to queue this action or execute it immediately |
| Repeat | Whether or not to issue this action multiple times |
| Delay | The number of game time-steps to wait until receiving the next observation |



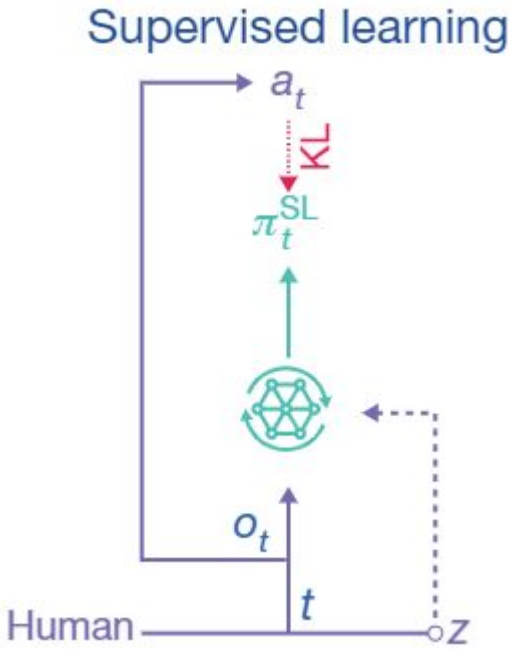



**Extended Data Fig. 3 | Overview of the architecture of AlphaStar.** A detailed description is provided in the Supplementary Data, Detailed Architecture.

**f** Architectures

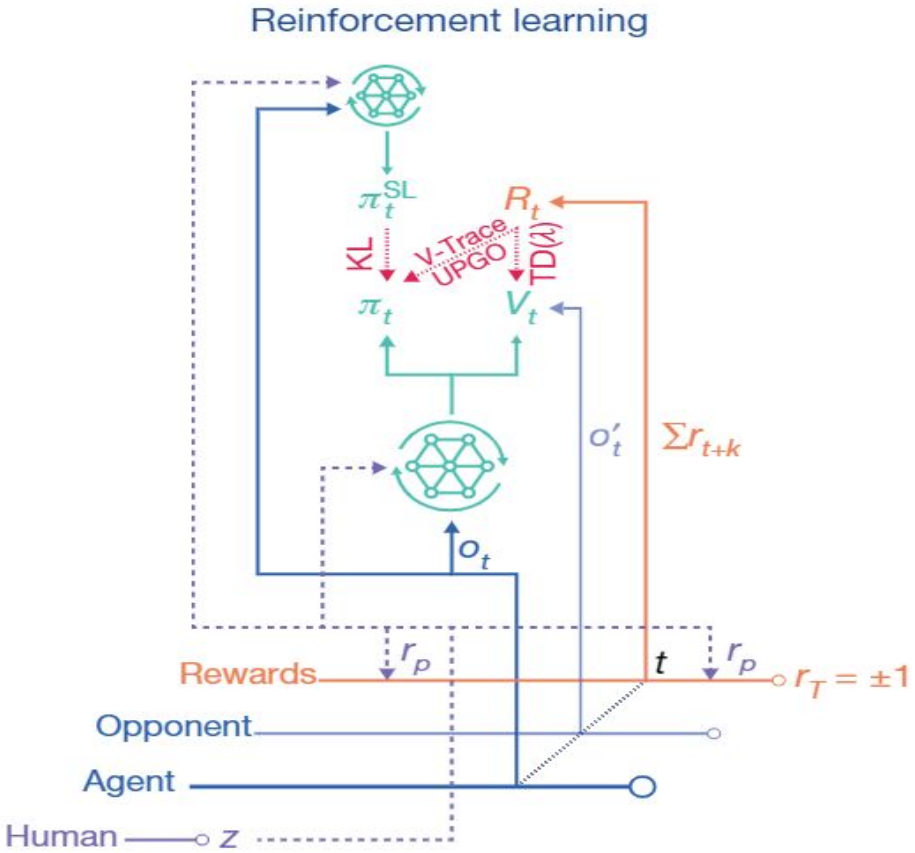


# Details : SL

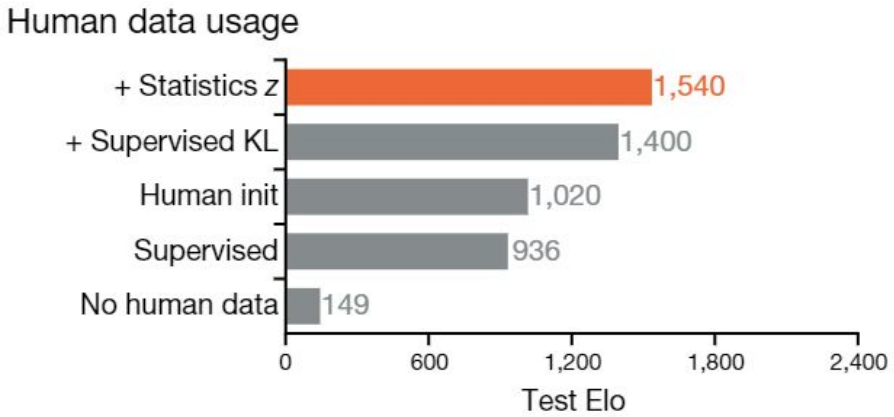



1. 971,000 replays (MMR > 3,500 : top 22% of player)
2. Loss : policy's output과 실제 사람의 action간 KL divergence. apply L2 reg, Adam optimizer
3. 각 replay마다 z를 추출 (최초 만든 유닛 및 건물의 20개로 판단) SL에서는 10%의 학습시간동안 z = 0
4. MMR 6,200유저의 승리 replay 16,000개를 사용해 policy를 fine-tune. => elite bot상대로 승률이 87%=> 96%
5. SL+fine tune agent는 MMR (Terran : 3,947, Protoss : 3,607 Zerg : 3,544)

# Details : RL

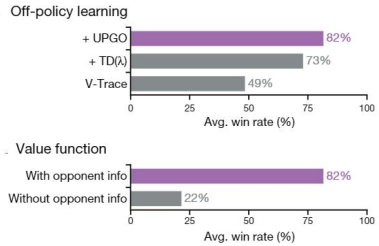



1. reward : 승 1, 무 0, 패 -1. no discount
2. agent vs agent
3. actor-critic $V_\theta(s_t, z)$ train => r 예측, policy update
4. SL로 policy 초기화
5. SL policy와 현재 policy간 KL div가 loss에 반영.
6. pseudo-reward : sample된 z를 따르도록
   pseudo-reward 별로 value function과 loss가 있음.
   25%로 발동



# Exploration

1. SL policy와 현재 policy간의 KL divegence를 줄이도록 한다.
2. Main agent를 human data로 부터 추출된 z를 따르도록 pseudo-reward로 학습. 실제 build와 z의 build간의 Hamming distance를 사용.




# Value & Policy update

1. Actor에 의해 생성된 trajectory를 replay buffer에 넣고, 비동기적으로 Learner가 파라미터를 업데이트 => Need off-policy correction
2. off-policy correction for policy(for stability) : V-trace + uncorrected update of the value func(bias 추가 되지만 variance를 줄임) : TD(lambda)
3. 각 action type이 독립적이라고 가정 + opponent's observation(for value only training)
4. policy update에는 UPGO사용.
5. 총체적 loss는 weighted sum (policy, value, reward, pseudo reward, reg)
6. 요걸 Adam에..

clipped importance ratio

$$\rho_t(G_t^U - V_\theta(s_t, z))\nabla_\theta \log \pi_\theta(a_t \mid s_t, z) \qquad \rho_t = \min\left(\frac{\pi_\theta(a_t \mid s_t, z)}{\pi_{\theta'}(a_t \mid s_t, z)}, 1\right)$$

$$G_t^U = \begin{cases} r_t + G_{t+1}^U & \text{if } Q(s_{t+1}, a_{t+1}, z) \geq V_\theta(s_{t+1}, z) \\ r_t + V_\theta(s_{t+1}, z) & \text{otherwise} \end{cases}$$

one step target

$$Q(s_t, a_t, z) = r_t + V_\theta(s_{t+1}, z)$$

# League training =: Multi agent learning

1. Self play시 발생되는 cycle문제 해결과 다양한 전략에 대한 통합적 통찰
2. 리그의 상대방 매칭 알고리즘으로서 기존의 FSP(Fictitious Self Play)에서 게임이 낭비되는 걸 막고자 PFSP(Prioritized~)사용. => 내가 가장 도전할만한 상대골라줌.
3. A가 후보군 C에서 B를 뽑을 확률

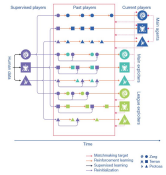
Main agent : 35% SP, 50% PFSP, 15% PFSP in forgotten player and past main exploiters. 만약 질놈이 없으면 15%는 SP

$$\frac{f(\mathbb{P}[A \text{ beats } B])}{\sum_{C \in \mathcal{C}} f(\mathbb{P}[A \text{ beats } C])}$$

$f: [0,1] \to [0,\infty)$ is some weighting function

$$f_{\text{hard}}(x) = (1-x)^p$$

$$f_{\text{var}}(x) = x(1-x)$$



# League training =: Multi agent learning

Main agent : 우리가 실제 내놓을 놈. 35% SP, 50% PFSP, 15% PFSP in forgotten player and past main exploiters. 만약 질놈이 없으면 15%는 SP. 20억 스텝마다 Main agent의 copy가 리그에 투입.

| 50% PFSP | 35% SP | 15% conditioned PFSP |
|---|---|---|

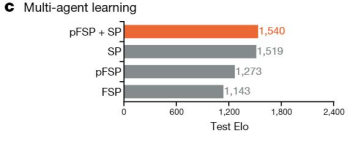
League exploiter : 리그 분석가. 승률 70% 이상되거나 20억스텝마다 추가. 추가될때 25%로 SL로 초기화

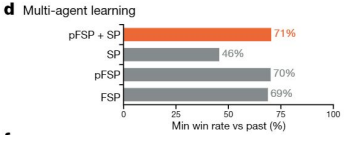
| 100% PFSP |
|---|

Main exploiter : Main main agent와 PFSP $f_{\text{var}}(x) = x(1-x)$ 만약 진행기간중 절반이? 승률이 20% 밑으로 떨어지면, 과거의 3종족 main agent에 대해 승률이 70%이상이거나 40억스텝뒤 초기화

| 100% PFSP vs Main agent. |
|---|

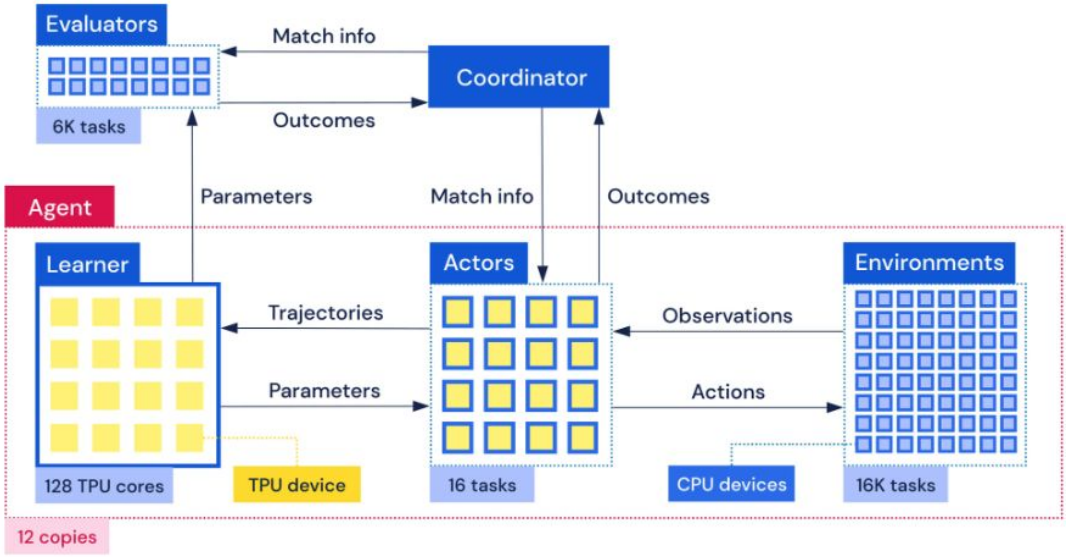
종족별 1,1,2개로 시작.

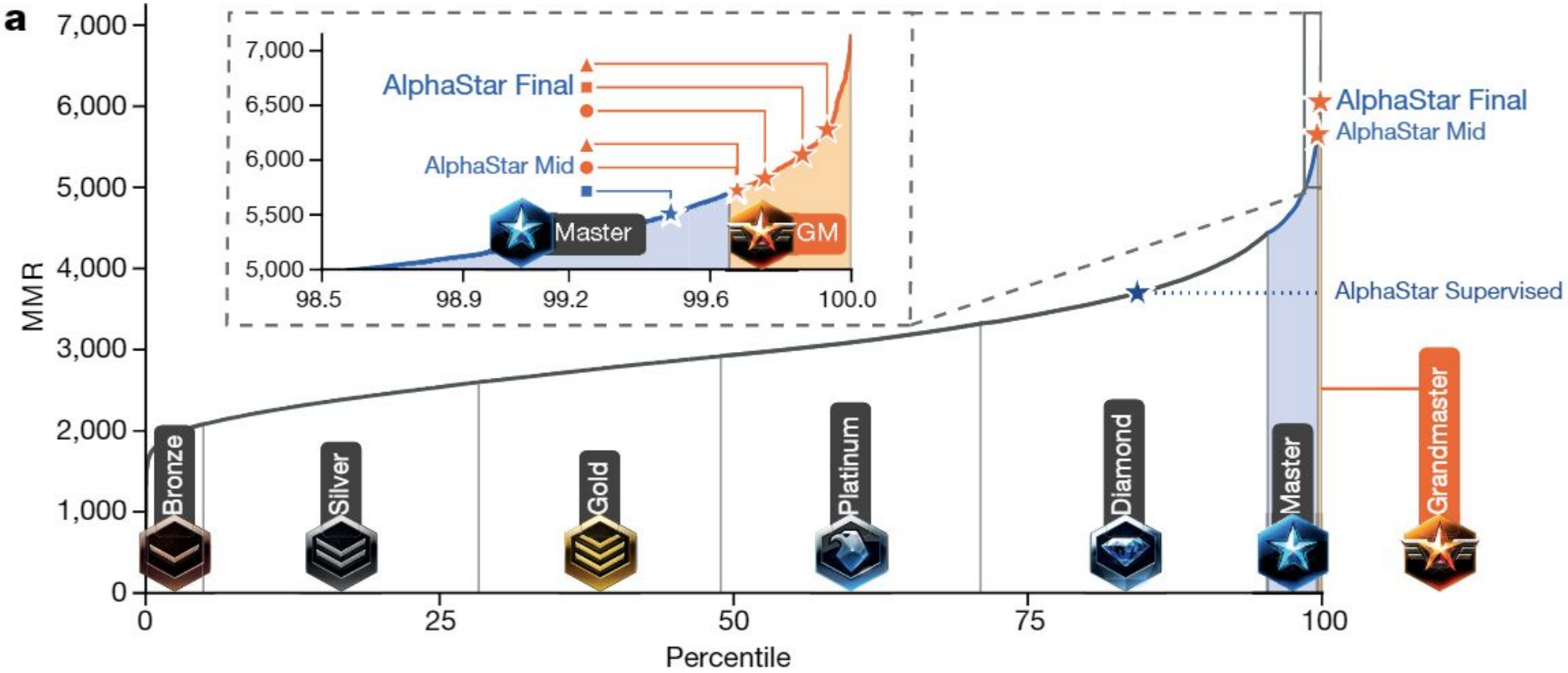


# Infra structure

1. 12개의 agent. 각자 16,000개의 게임을 돌리는 중
2. 16개의 actor task. task당 8 TPU core
3. Learner의 각 core당 4개의 trajectory sequence mini batch.
4. 10초마다 Actor의 파라미터 업데이트.






- AlphaStar Supervised : supervised training only, 30 games from unranked, 상위 16% 도달(3699 mmr)
- AlphaStar Mid : 27 days of league training, 60 games from unranked, 상위 0.5% 도달
- AlphaStar Final : 44 days of league training, 30 games from Mid, 상위 0.15% 도달(6275 mmr)

# 소감 및 결론

1. 엥? 각 unit들이 학습해서 Multi agent인거 아니였어?
2. 엥? 사람데이터 썼어?
3. z를 써서 domain 지식을 함축하는 건 좋은 아이디어 같음.
4. 복잡한 문제를 복잡하게 풀었다.