논문 발표 - 2

Grandmaster level in StarCraft II using multi-agent reinforcement learning

Being top 0.02% against Human Players

이름	신재영
학부	전산학부

00 What is StarCraft?

- Involves high-level strategic decisions and individual control of units.
- It features three distinct races: Terran, Protoss, and Zerg.
- Players start with a small base and worker units to gather resources, build additional units, scout opponents, and research technologies.
- The goal is to defeat the opponent by eliminating their buildings.

01 Why StarCraft?

AlphaGo

AlphaStar

01 Why StarCraft?

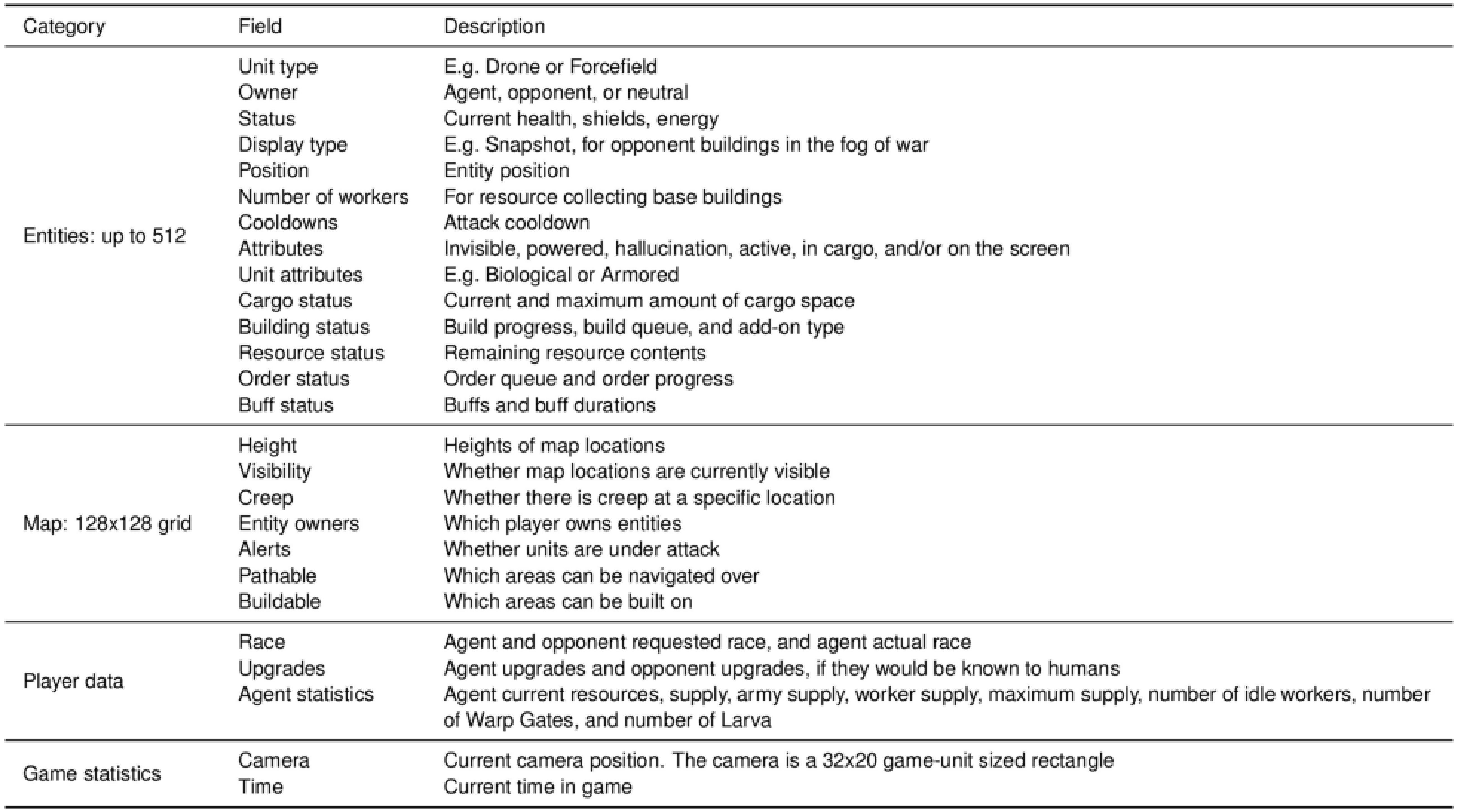
Go

- 371 action spaces
- All the information are given
- 1 player combination (only 1 vs 1)
- Simple grid environment

StarCraft 2

- 10^26 action space
- Some of the information is hidden depending on camera's view
- 6 player combinations (3 different species)
- Very complex environment
    - camera movement
    - constructing buildings
    - controlling individual troops
    - so on...

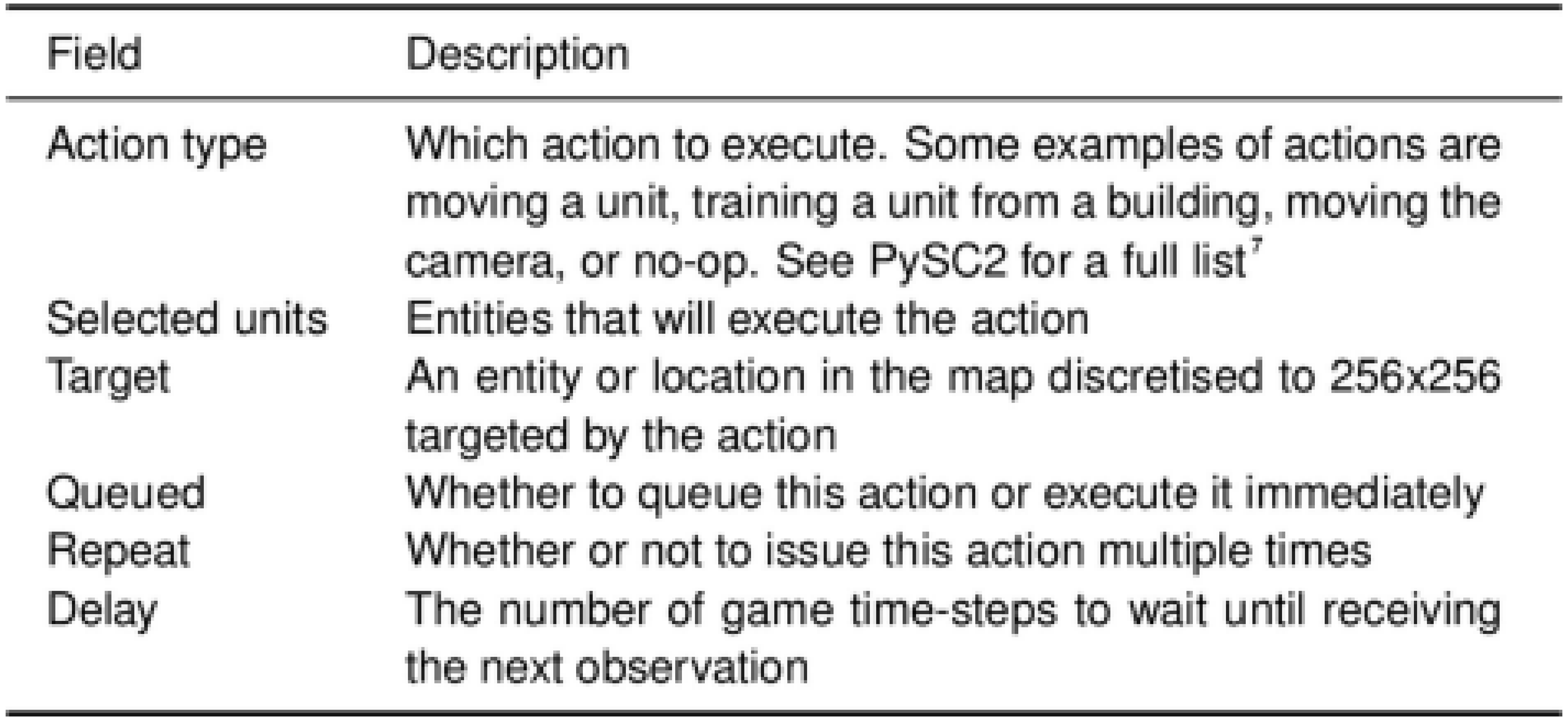
02 Agent Input Space



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

Too Many and diverse Input Space

03 Agent Action Space



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

Very complicated action with location and timing taken into account

04 Policy in AlphaStar

$$\pi_\theta(a_t \mid s_t, z) = \mathbb{P}[a_t \mid s_t, z]$$

$$s_t = (o_{1:t}, a_{1:t-1})$$

- The policy is implemented as a deep neural network.
- Inputs include all previous observations and actions
- The policy is also conditioned on a strategy statistic z
  - includes the sequence of the first 20 constructed buildings and units in a game
  - captures the initial strategic decisions made by players
  - include the units, buildings, effects, and upgrades present during the game
  - By leveraging human expertise encoded in z, inefficient trial-and-error phase for basic strategies can be skipped

05 Architecture - Observation

Spatial information = feature map (similar to CNN)
Unit Attributes = feature vectors

Self-Attention Mechanism: used to process the relationships between units and their attributes, allowing the model to focus on different parts of the observation dynamically.

LSTM Networks: used to process the sequence of observations over time, capturing the temporal dependencies and dynamics of the game state.

- input to the LSTM includes the current and past observations, providing the context needed to make informed decisions.

05 Architecture - Action

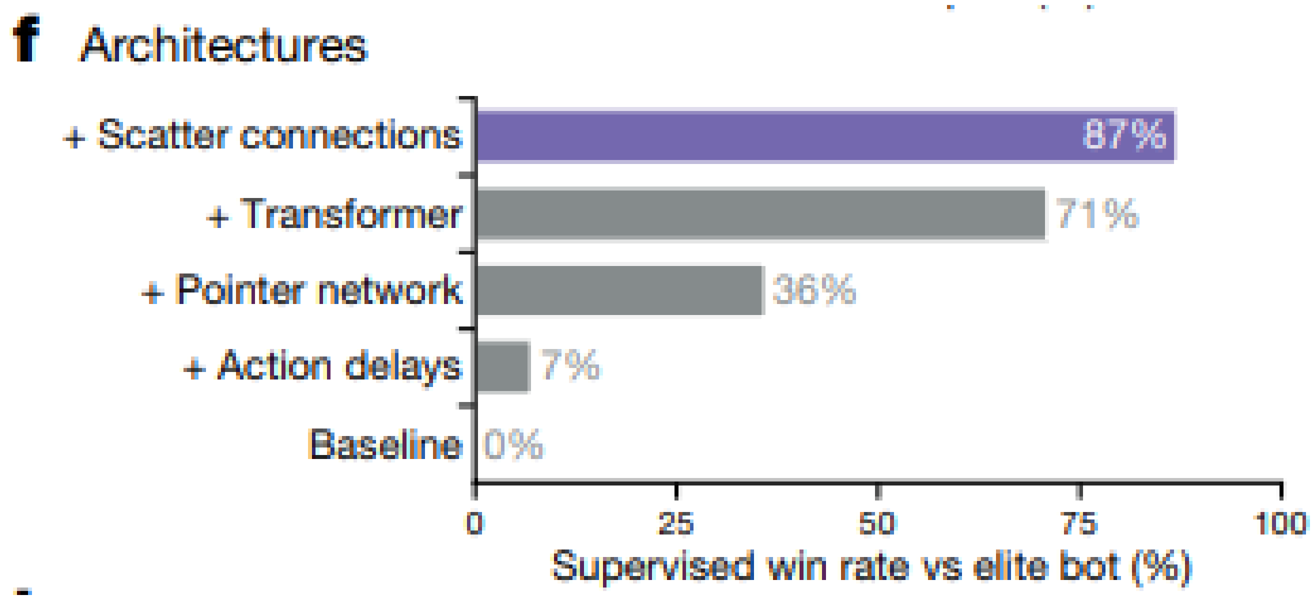
Hierarchical Action Representation:

- Action Type: select the type of action (e.g., move, attack, build).
- Action Parameters: For each action type, the relevant parameters are selected, such as which unit to move, the target position, or the building type.
- Auto-Regressive Policy: The action parameters are generated in a sequence, conditioned on the previous parameters and the current observation.

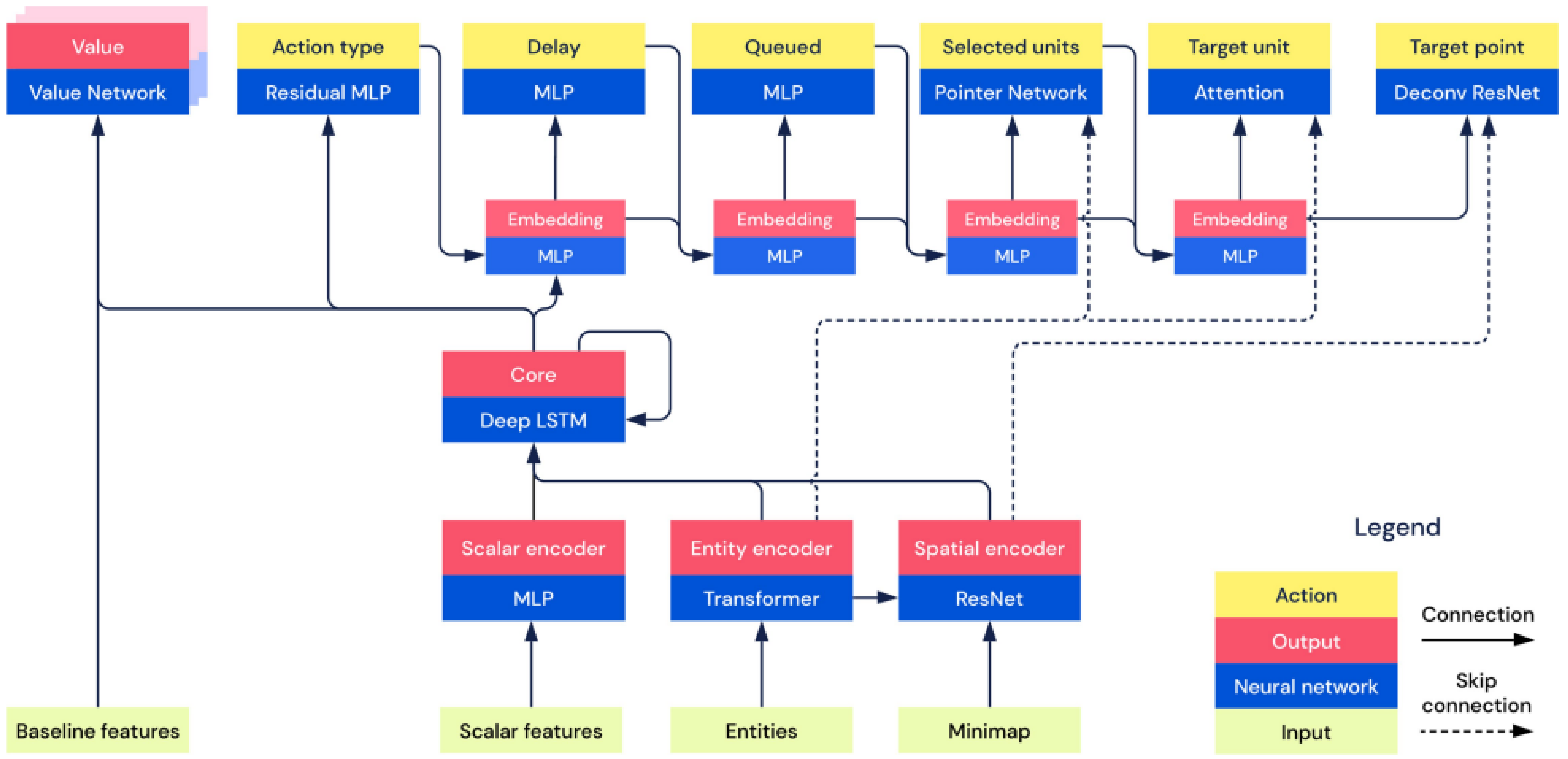
Recurrent Pointer Network: For actions requiring target selection

Scatter Connections: Integration of Spatial and Non-Spatial Information

- a type of connection that allows a vector to be "scattered" onto a layer representing a map, so that a vector at a specific location corresponds to objects of interest at that location
- used to integrate spatial (e.g., positions on the map) and non-spatial information (e.g., unit attributes) in the action encoding process.

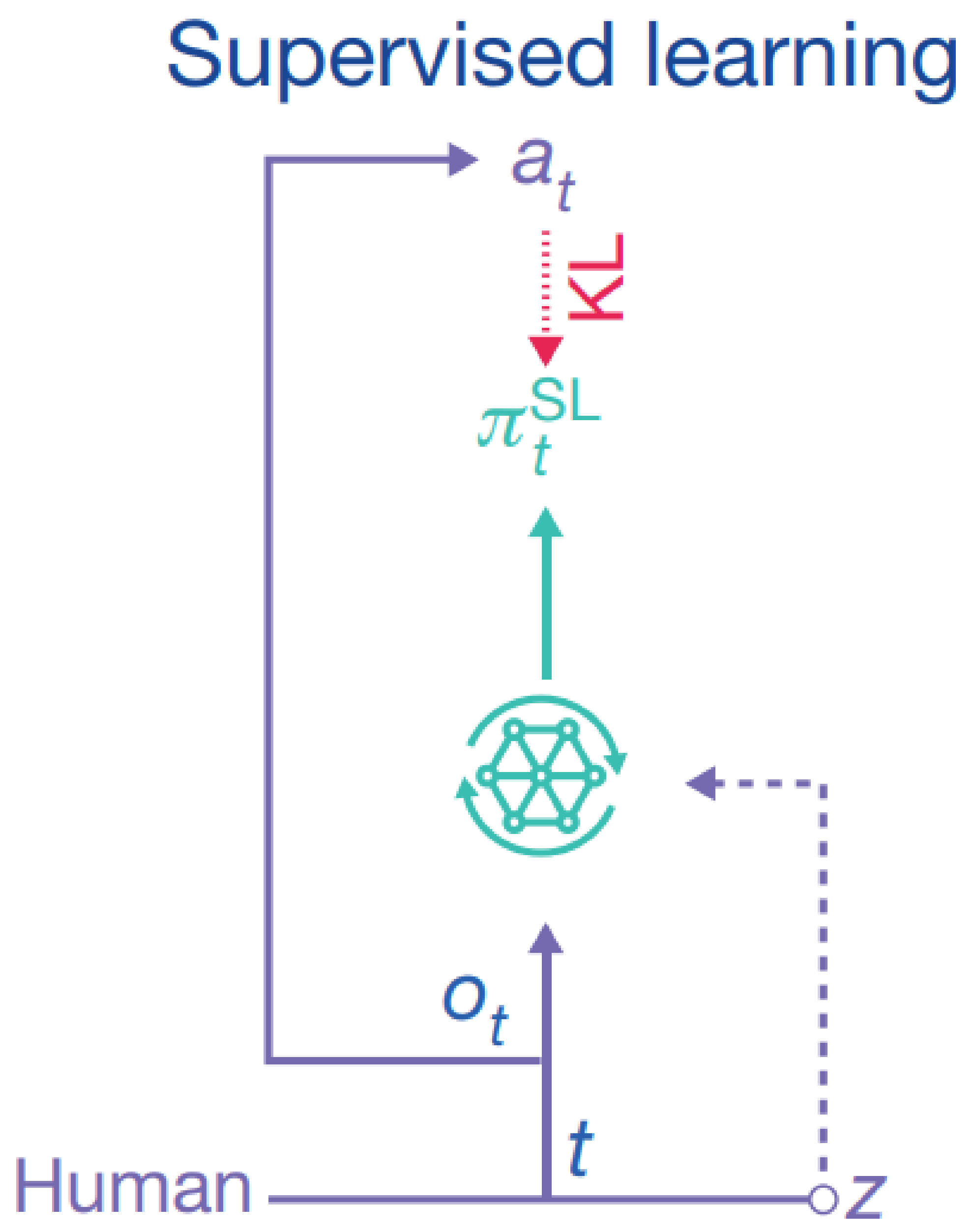


05 Architecture



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

06 Supervised Learning in AlphaStar



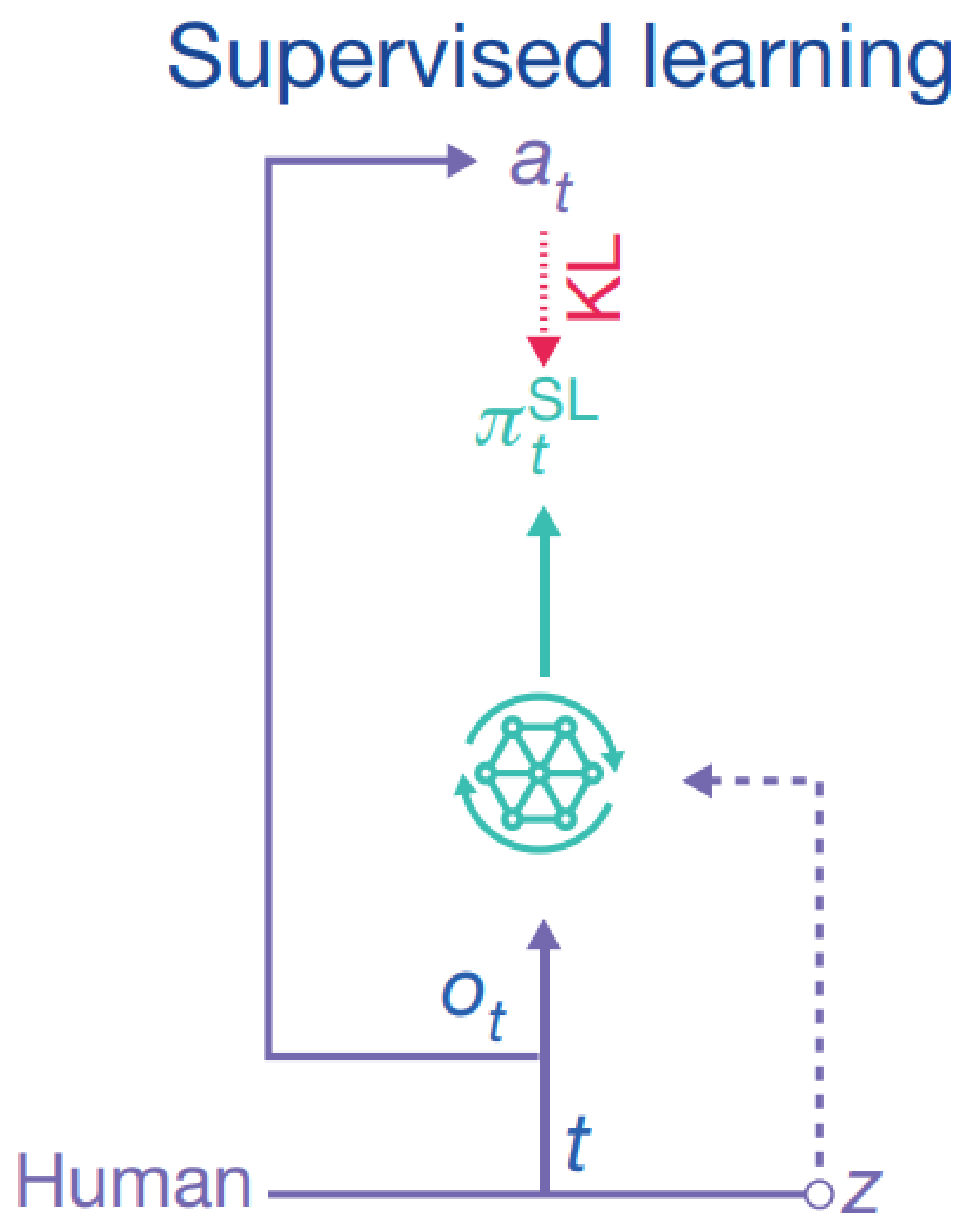
First trained using human data to improve efficiency

- imitation learning

Dataset: 971,000 replays played on StarCraft II by players with MMR (Match Making Rating) scores above 3,500 (top 22% of players)

Policy is initialized by training a neural network to predict the actions taken by human players given the same game state

06 Supervised Learning in AlphaStar - Training



$$D_{KL}(p(x)\,||\,q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)}$$

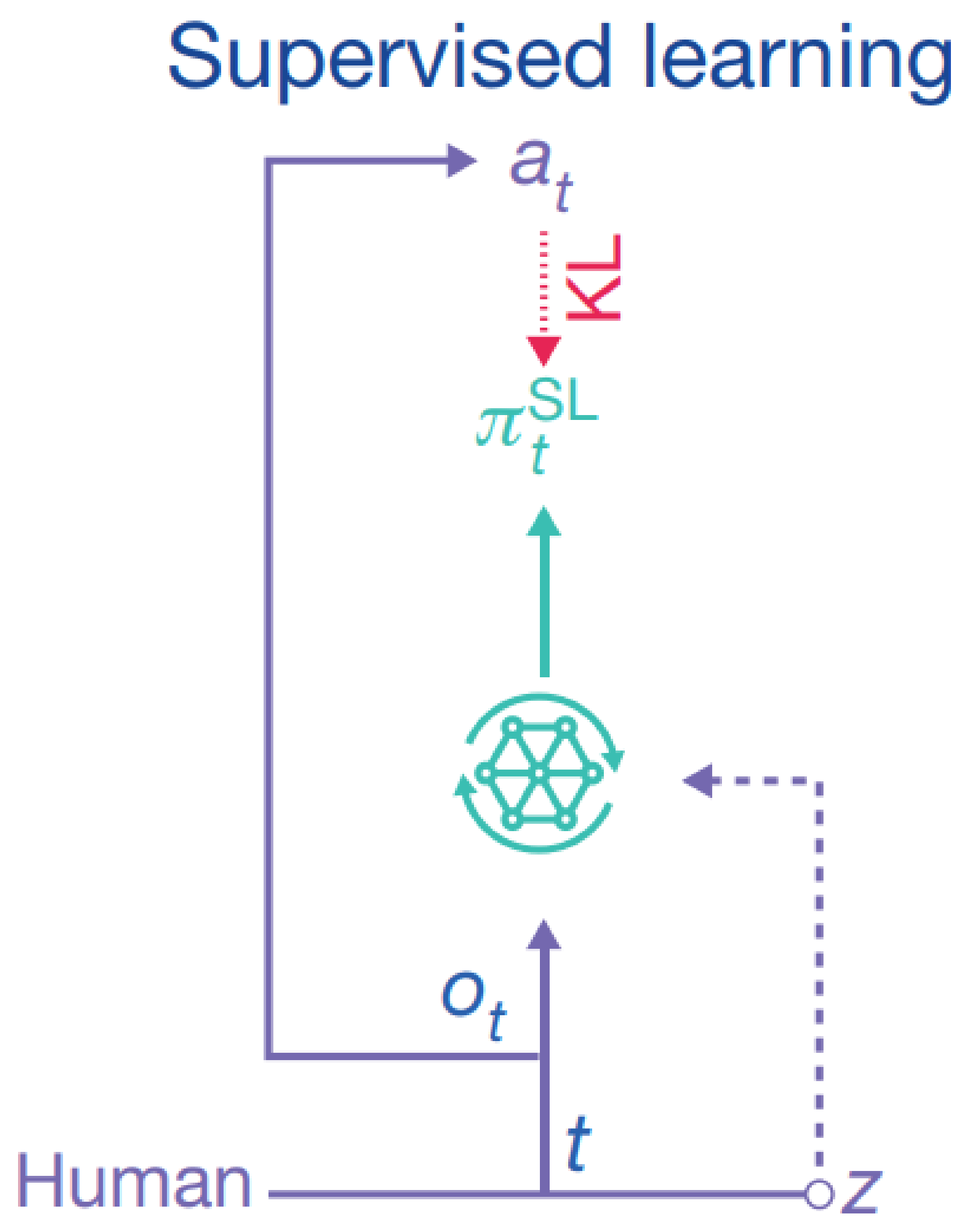
Outputs a probability distribution over possible actions

KL divergence is used to calculate difference between probability distribution of human and agent actions
Optimization:

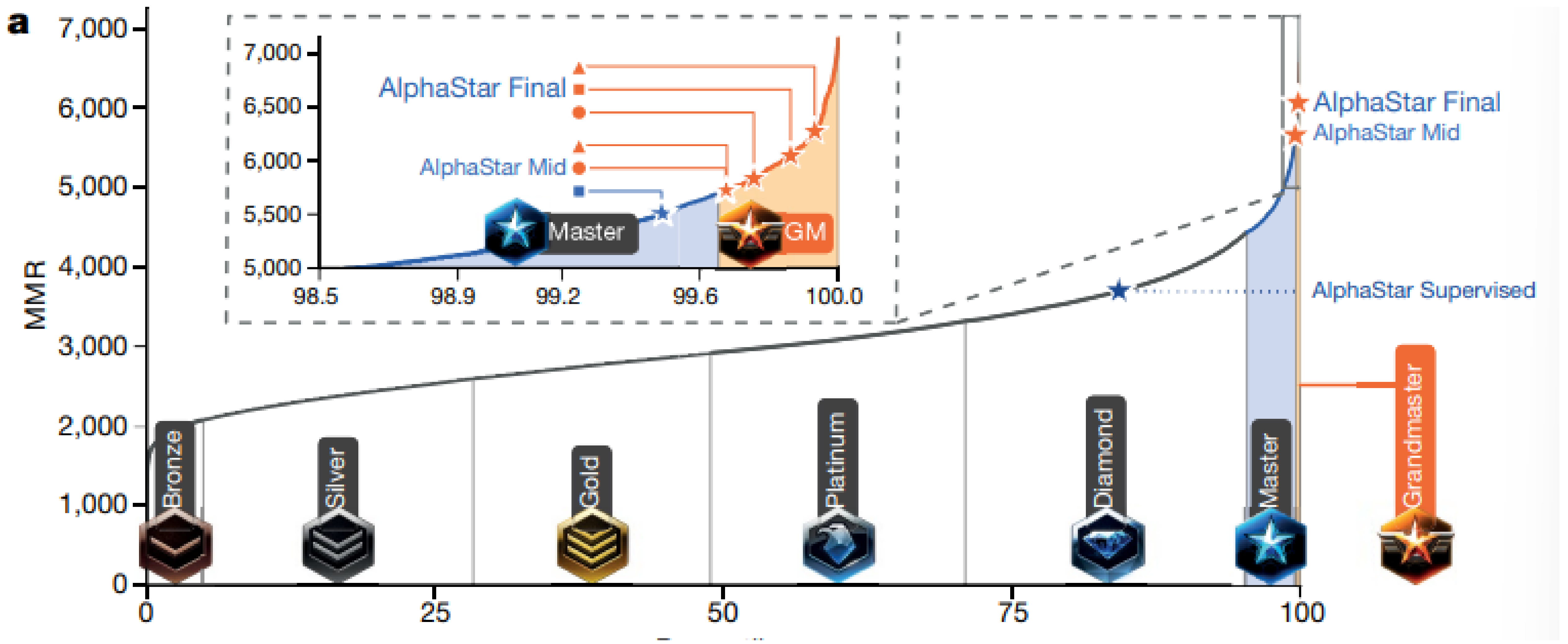
Adam optimizer used with L2 regularization

$$\text{L2:} \quad R(\theta) = ||\theta||_2^2 = \sum_{i=1}^{n} \theta_i^2$$

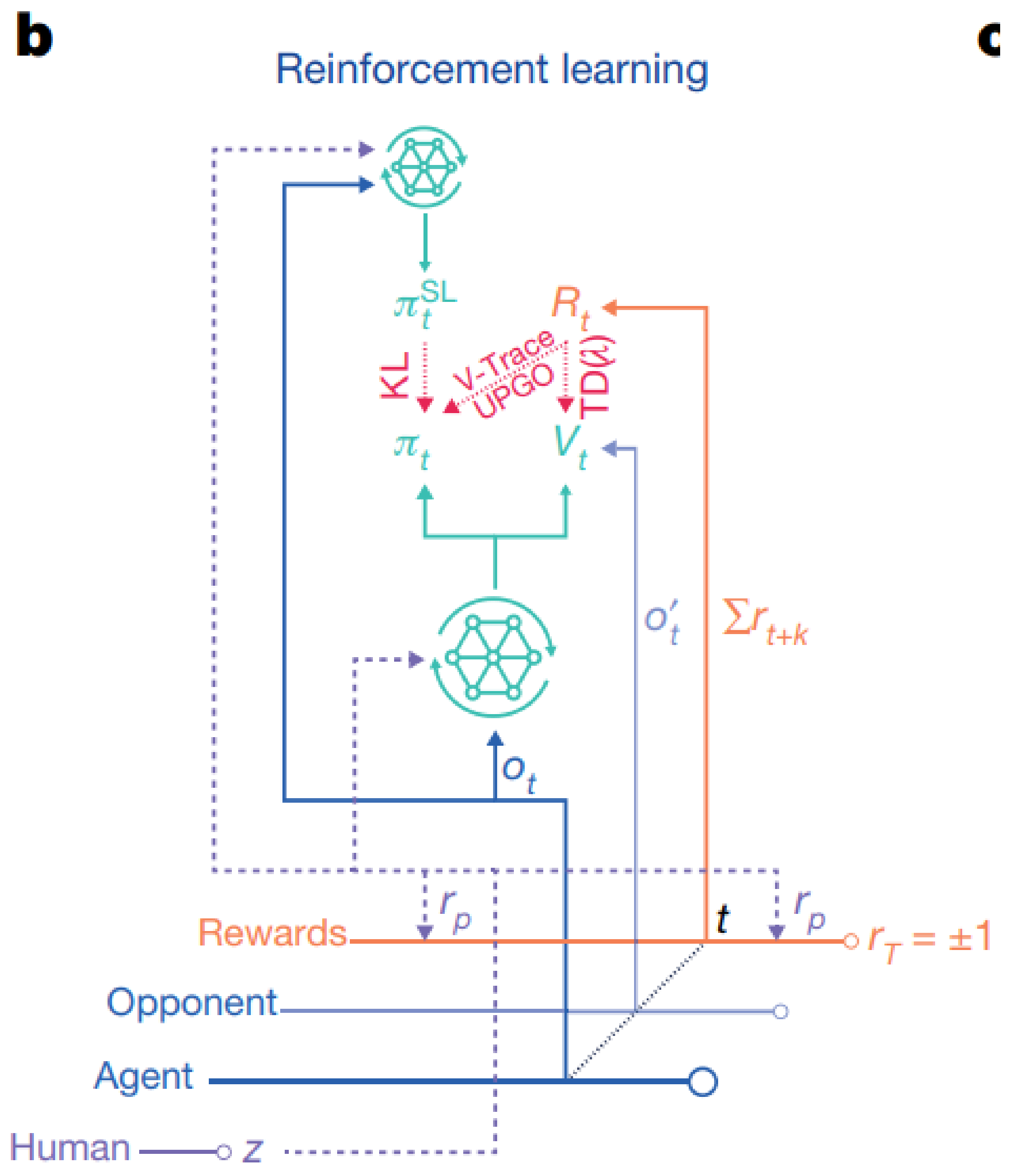
06 Supervised Learning in AlphaStar - Fine Tuning



Fine-tuned using only winning replays with MMR above 6,200, further enhancing the agent's ability to execute high-level strategies.



07 Reinforcement Learning in AlphaStar



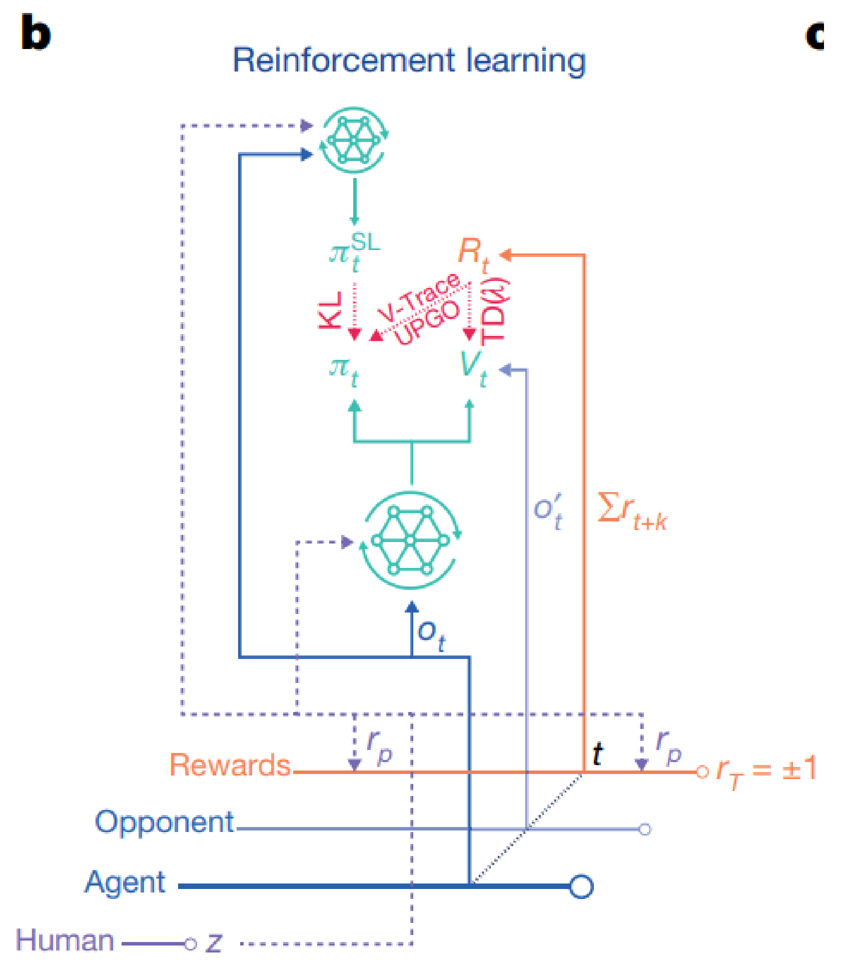
The initialized policy is then further trained using self-play and league training.

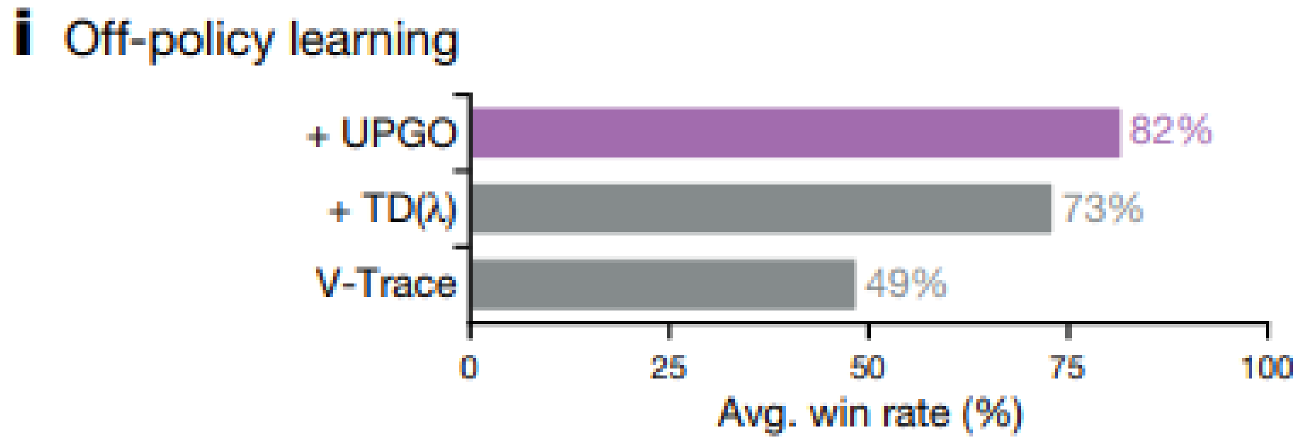
Reward: +1 for win, 0 for draw, and -1 for loss

Actor-critic based design is used

- policy-based (actor) and value-based (critic) methods
  - actor: updates the policy parameters in the direction that improves the expected return
  - critic: feedback to the actor on how good the selected action is, compared to the average.

07 Reinforcement Learning in AlphaStar





Temporal-Difference Learning: used to update value function based on the difference between predicted and observed rewards.

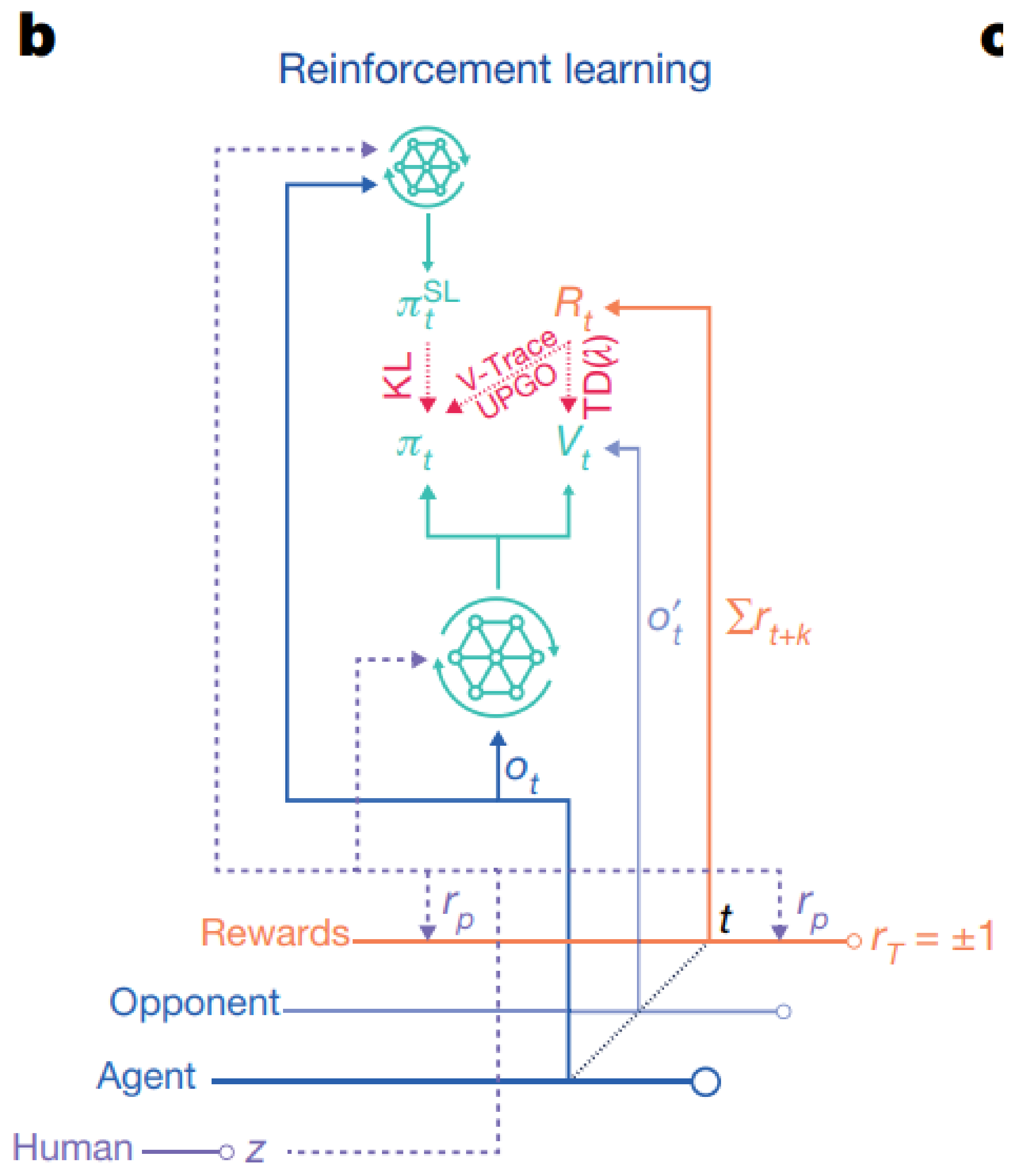
V-trace: an off-policy correction method that helps stabilize learning when using experience data generated by different policies.

- truncates the importance weights to control variance and prevent instability

UPGO (Upgoing Policy Update): focuses on trajectories with better-than-average returns, reinforcing successful strategies.

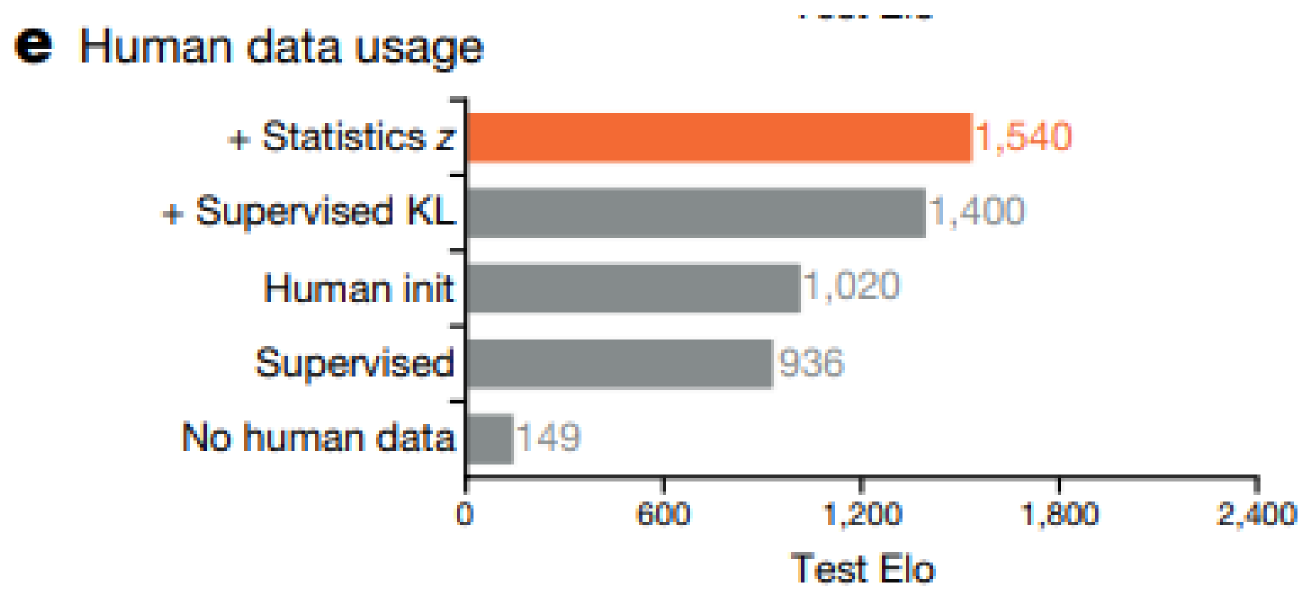
- modifies the policy update to give more weight to actions leading to higher rewards.

07 Reinforcement Learning in AlphaStar

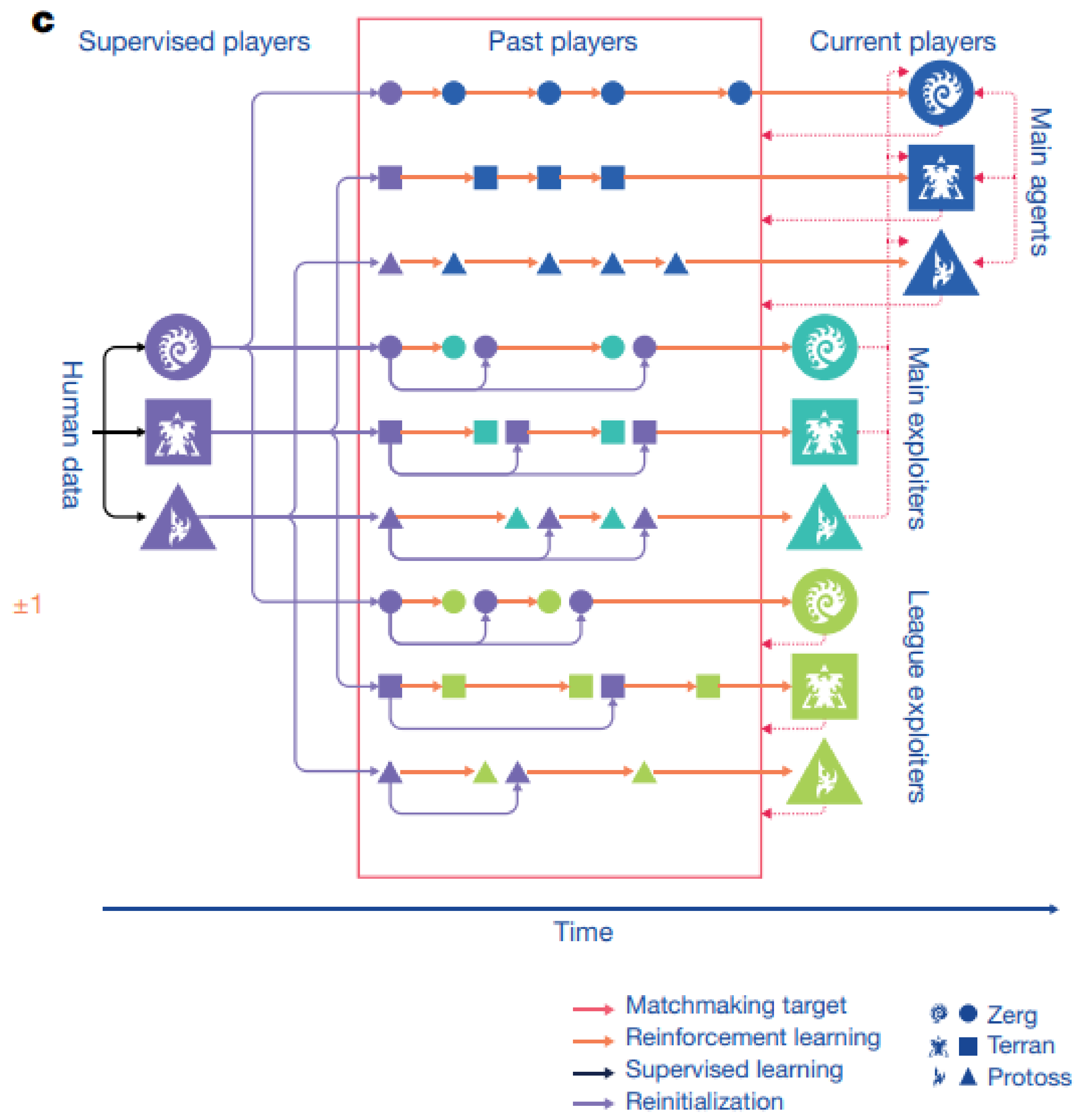


Kullback-Leibler (KL) divergence is minimized between the supervised policy and the current policy to retain useful human-like behaviors.

Agent also receives pseudo-reward by following strategy z.



07 Reinforcement Learning in AlphaStar

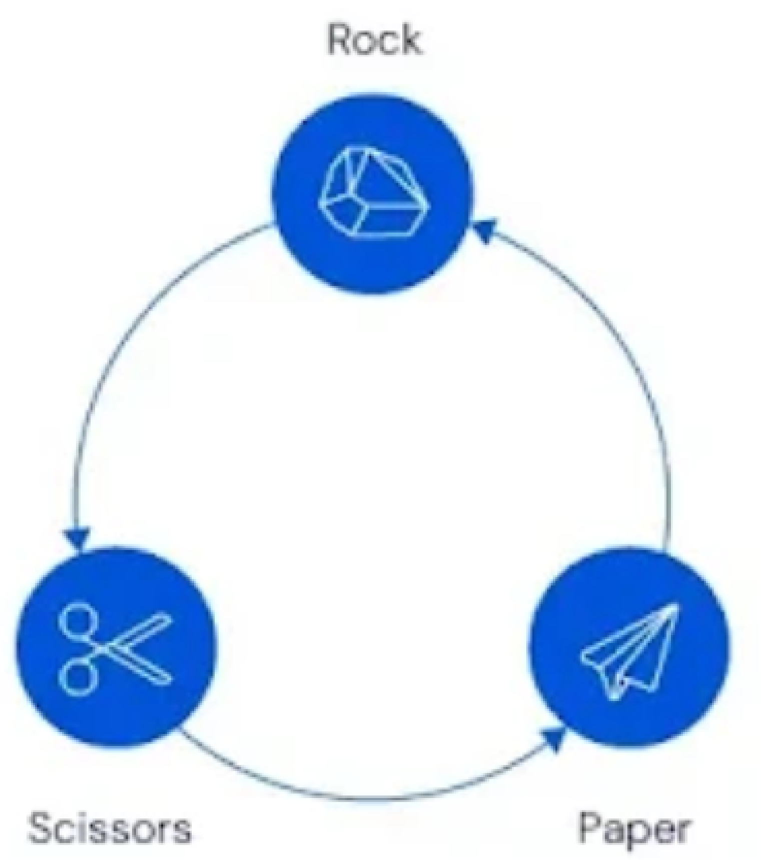


Multi-Agent Learning: competes against various versions of itself and other agents.

League Training: AlphaStar uses a league of agents to ensure diverse and robust training.

- **main agents**: learns strategies by playing against various opponents, including its past versions, other main agents, and exploiters
- **main exploiters**: accelerate the main agent's learning by presenting it with difficult and targeted challenges.
- **league exploiters**: exploit weaknesses not only in the main agent but in other agents within the league as well.

07 Reinforcement Learning in AlphaStar



Limitation of self-play:

1. Cycle of Strategies:

- can lead to cycles where the agent repeatedly switches between a set of strategies without progressing.
  - occurs when an agent learns a strategy to defeat its current opponent, then the opponent learns a counter-strategy, and the cycle continues without overall improvement.
- Ex) In StarCraft II, the agent might oscillate between aggressive rush tactics and defensive strategies without finding a stable, superior strategy.

2. Lack of Diversity:

- the agent predominantly learns to counter its own strategies.
- vulnerable to novel strategies that it has not encountered during training.

07 Reinforcement Learning in AlphaStar

Ficitious Self-Play(FSP): avoids cycle by playing with all players in the league

- too much waste of resources against agents with 100% win rate

Prioritized Ficitious Self-Play(PFSP): put weights on the match making in league to provide good learning signal

- using 2 different curriculum, it can improve by fighting with hard opponents and opponents around its level

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

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

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

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

07 Reinforcement Learning in AlphaStar

Main agent is trained on 35% SP, 50% PFSP against all past players in the league, and an additional 15% of PFSP matches against forgotten main players the agent can no longer beat and past main exploiters.

- copy of the main agent is added as new play in league every 2 billion steps

07 Reinforcement Learning in AlphaStar

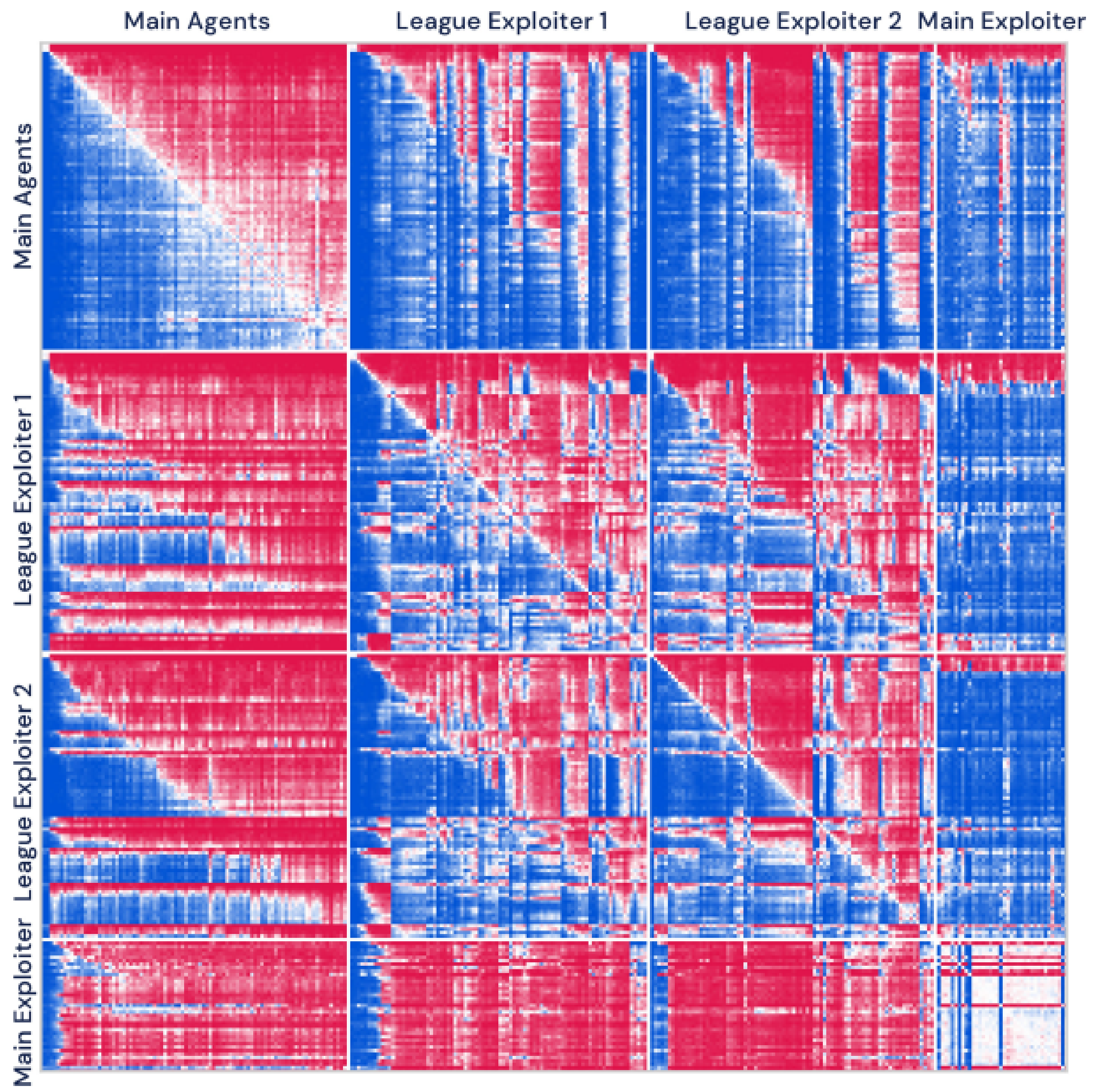
League exploiters are trained using PFSP and their frozen copies are added to the league when they defeat all players in the league in more than 70% of games, or after a timeout of 2 billion steps.

- a 25% probability that the agent is reset to the supervised parameters.
  - because league agents are not robust themselves

07 Reinforcement Learning in AlphaStar

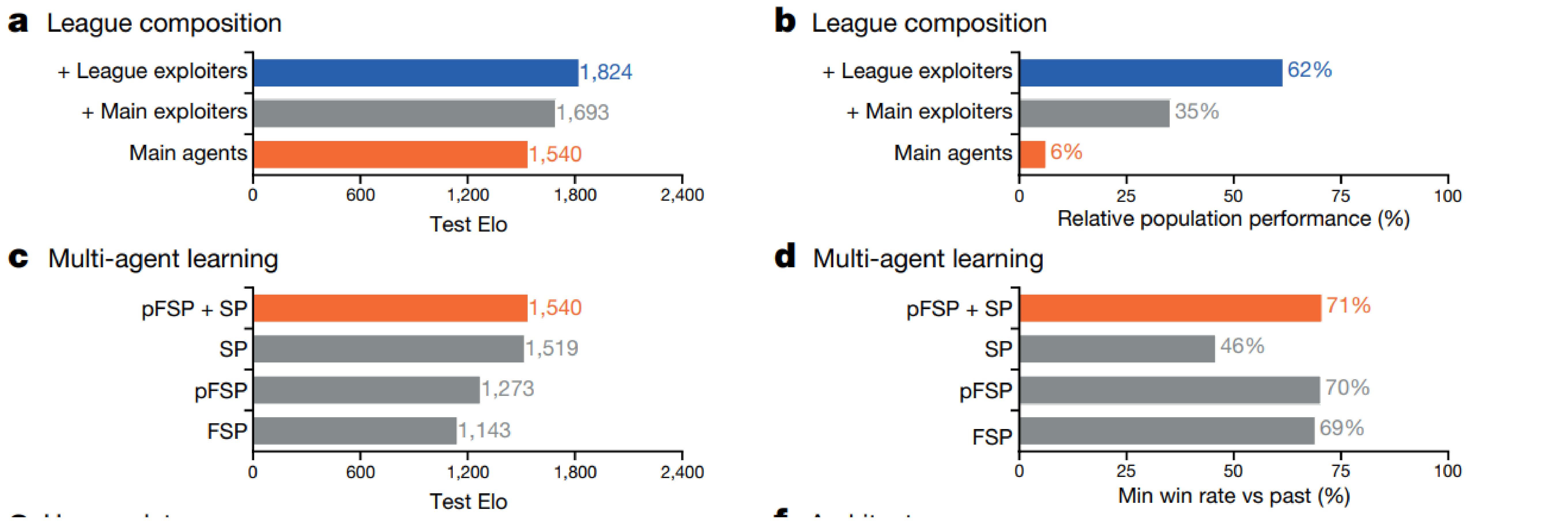
Main exploiters play against main agents.

- 50% of the time when the current probability of winning is lower than 20%, exploiters use PFSP with f_{var} weighting over players created by the main agents, forming curriculum that facilitates learning
- Otherwise there is enough learning signal and it plays against the current main agents. These agents are added to the league whenever all three main agents are defeated in more than 70% of games, or after a timeout of 4 billion steps, then reset to the supervised parameters.

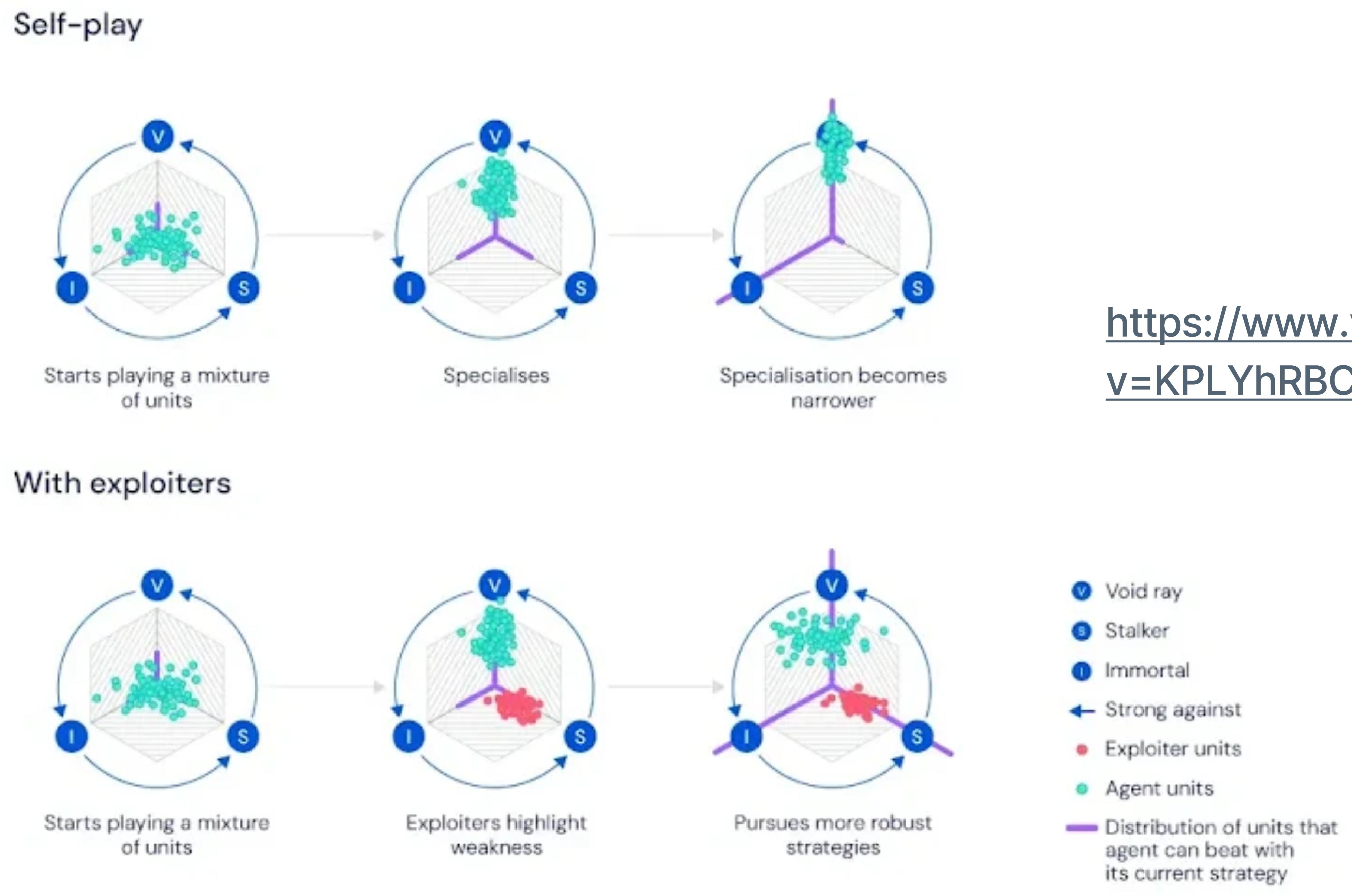


3,000,000 rock–paper–scissor cycles (with requirement of at least 70% win rates to form a cycle) that involve at least one exploiter, and around 200 that involve only main agents

07 Reinforcement Learning in AlphaStar



07 Reinforcement Learning in AlphaStar


Self-play
Starts playing a mixture of units
Specialises
Specialisation becomes narrower
With exploiters
Starts playing a mixture of units
Exploiters highlight weakness
Pursues more robust strategies
V
I
S
Void ray
Stalker
Immortal
Strong against
Exploiter units
Agent units
Distribution of units that agent can beat with its current strategy

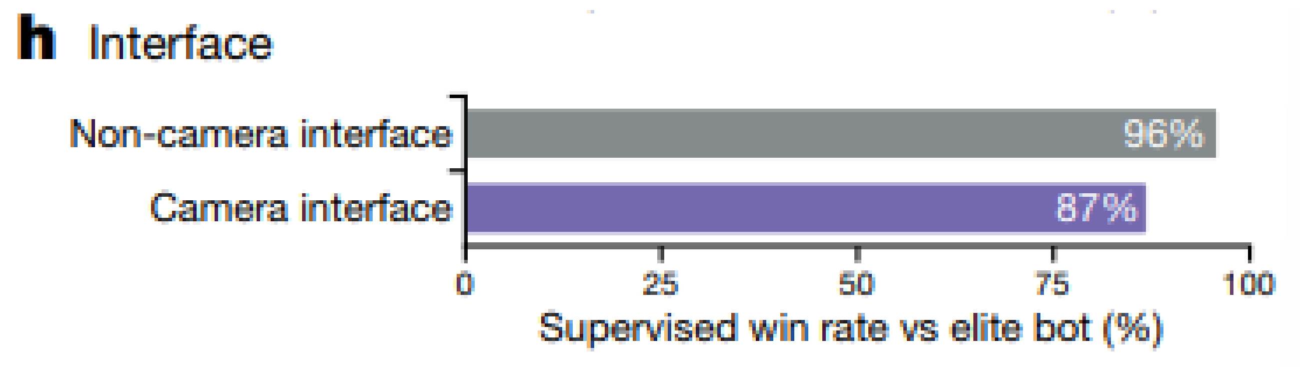
https://www.youtube.com/watch?v=KPLYhRBCcvk&t=76s&ab_channel=GoogleDeepMind

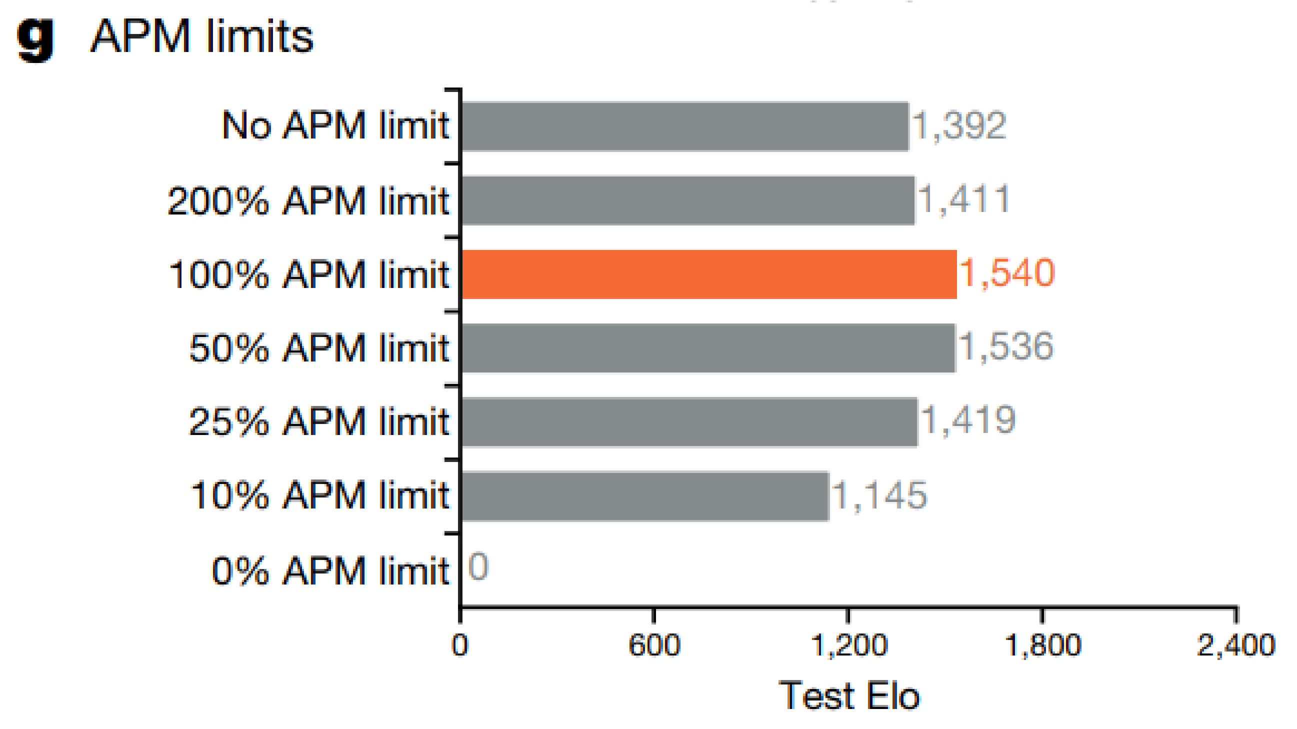
08 Restriction on Agents

1. Action per Minute (APM) Limits: executing at most 22 non-duplicate actions every 5 seconds.

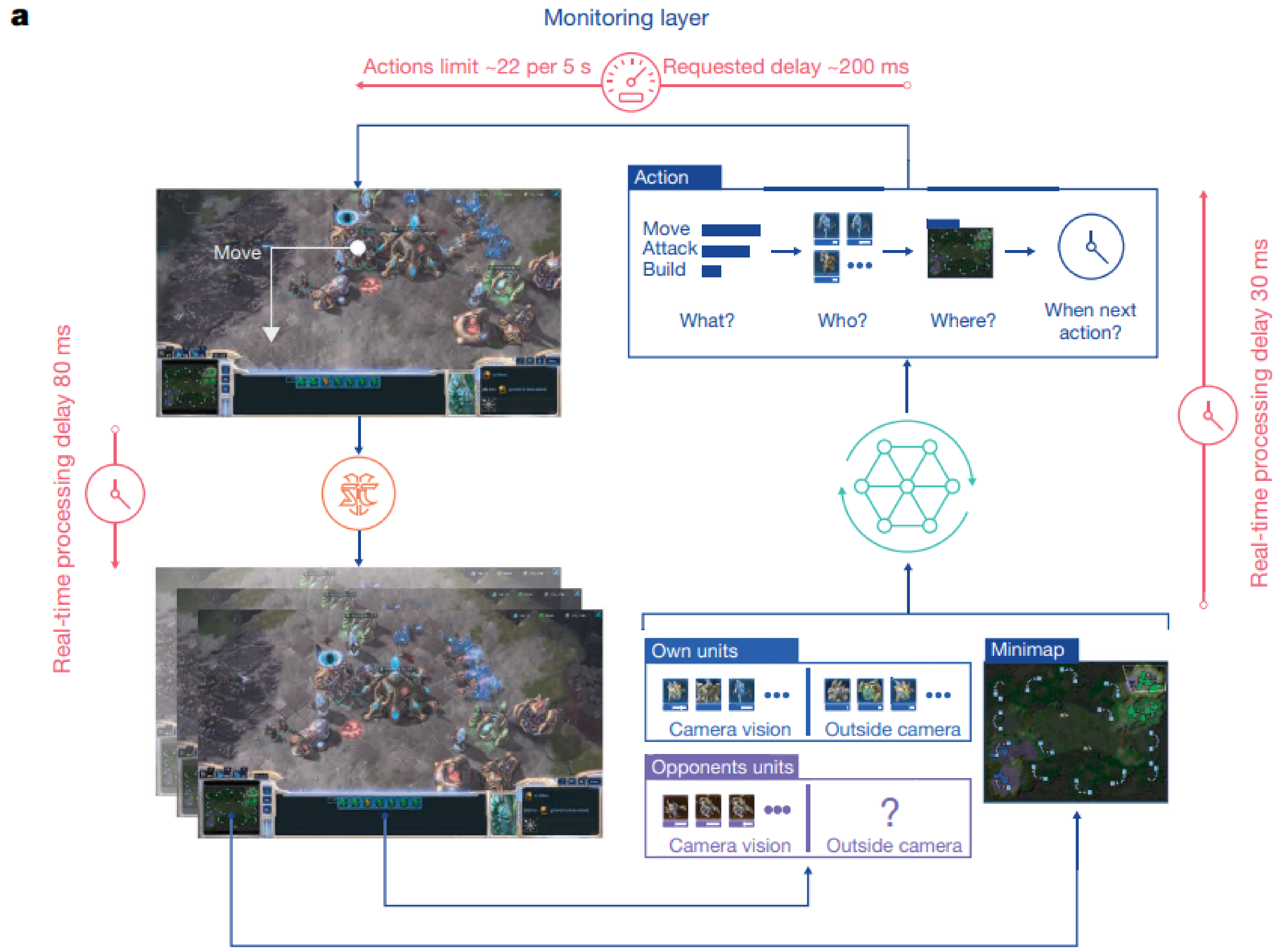
2. Delay in Actions: Human players experience a delay between perceiving the game state and executing an action.
- An average delay of about 110 milliseconds between command of action and actual execution
- Also, agents decide when to observe next (avg. 370ms)

3. Camera Interface: Human players control the game through a screen interface, limited by what they can see and interact with at any given time.
- could only act on information visible within the camera view.

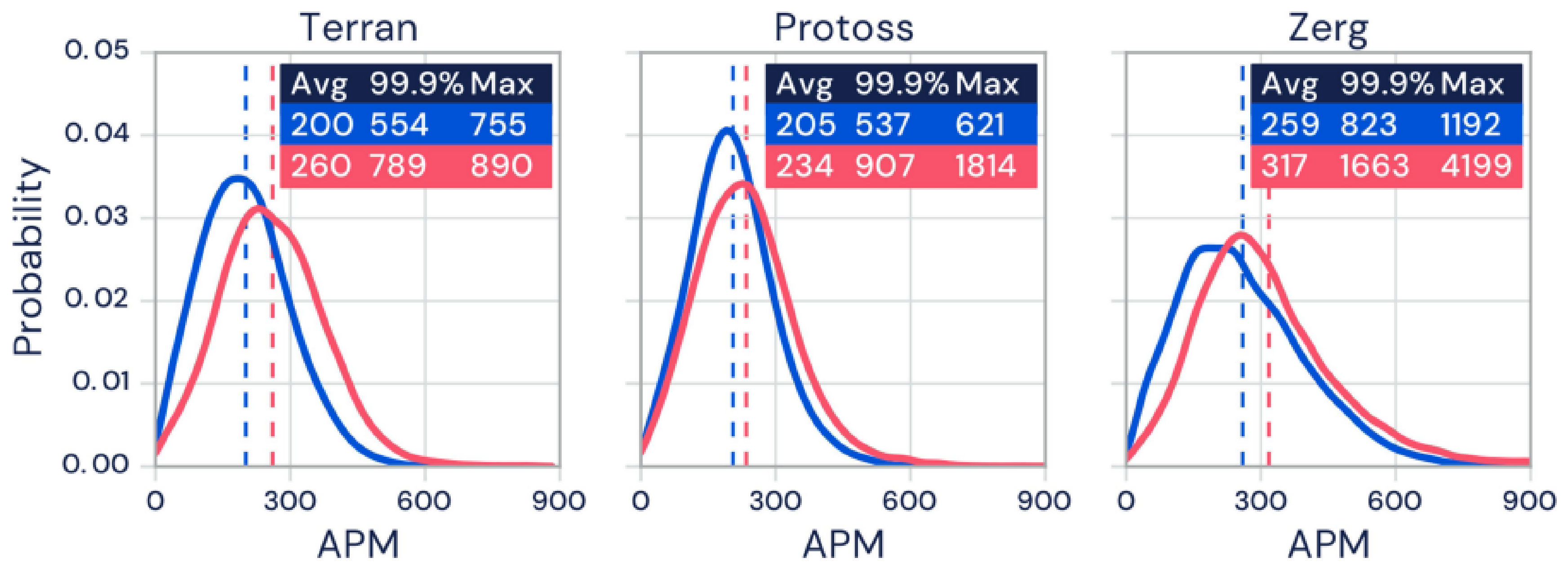




08 Restriction on Agents



08 Restriction on Agents

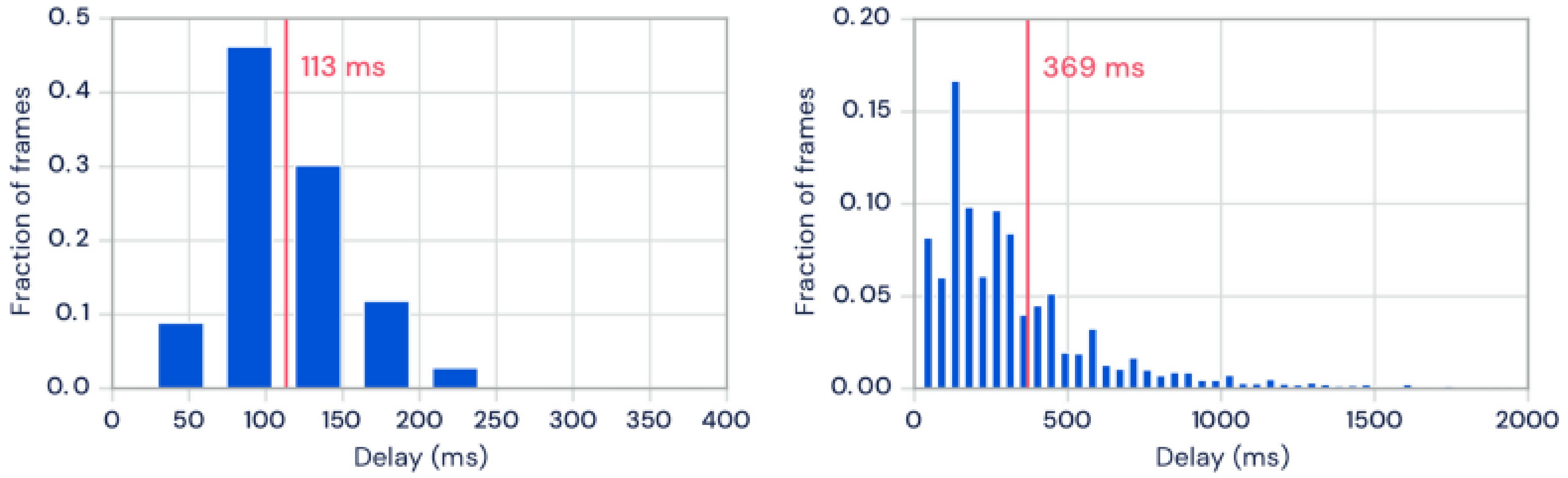


Terran Avg	99.9%	Max
200	554	755
260	789	890

Protoss Avg	99.9%	Max
205	537	621
234	907	1814

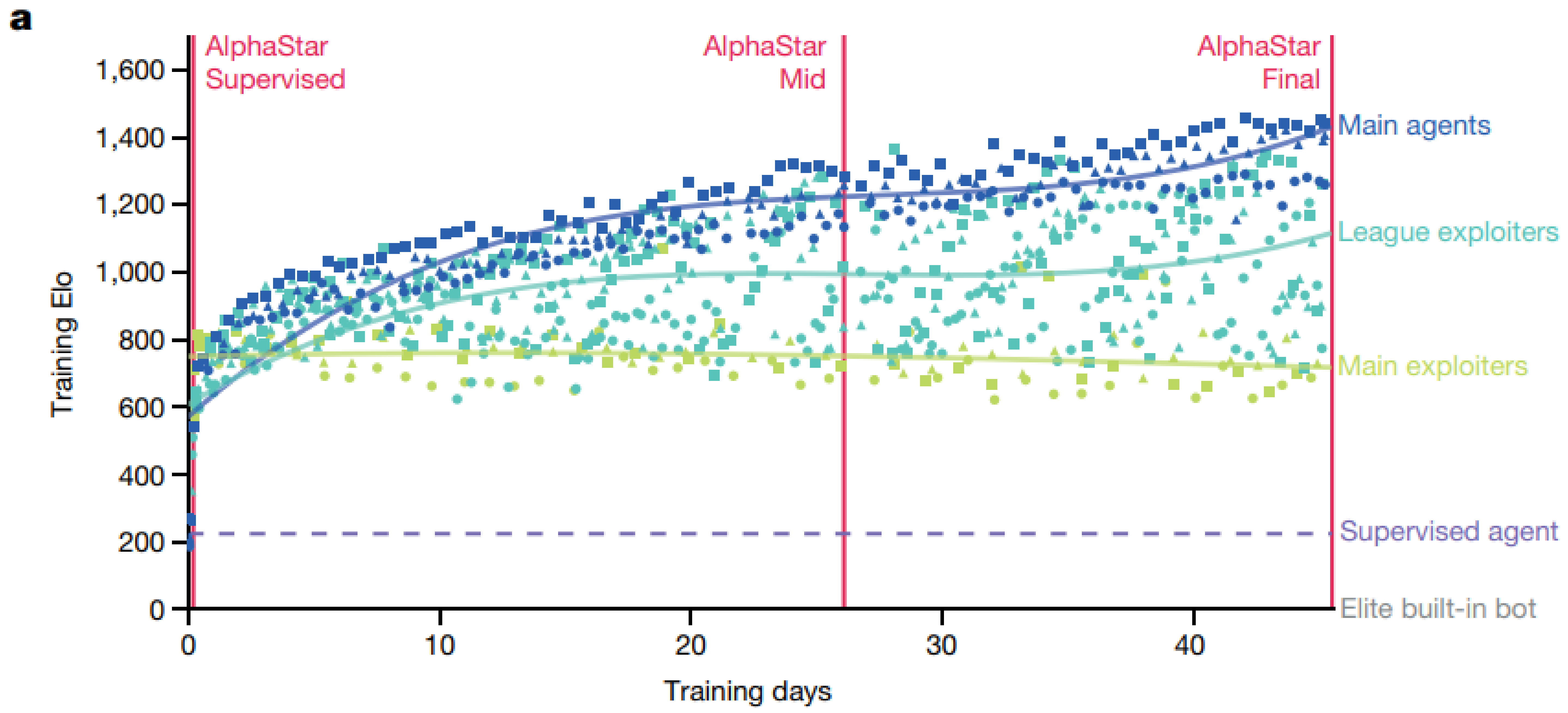
Zerg Avg	99.9%	Max
259	823	1192
317	1663	4199

Extended Data Fig. 1 | APM limits. Top, win probability of AlphaStar Supervised against itself, when applying various agent action rate limits. Our limit does not affect supervised performance and is acceptable when compared to humans. Bottom, distributions of APMs of AlphaStar Final (blue) and humans (red) during games on Battle.net. Dashed lines show mean values.

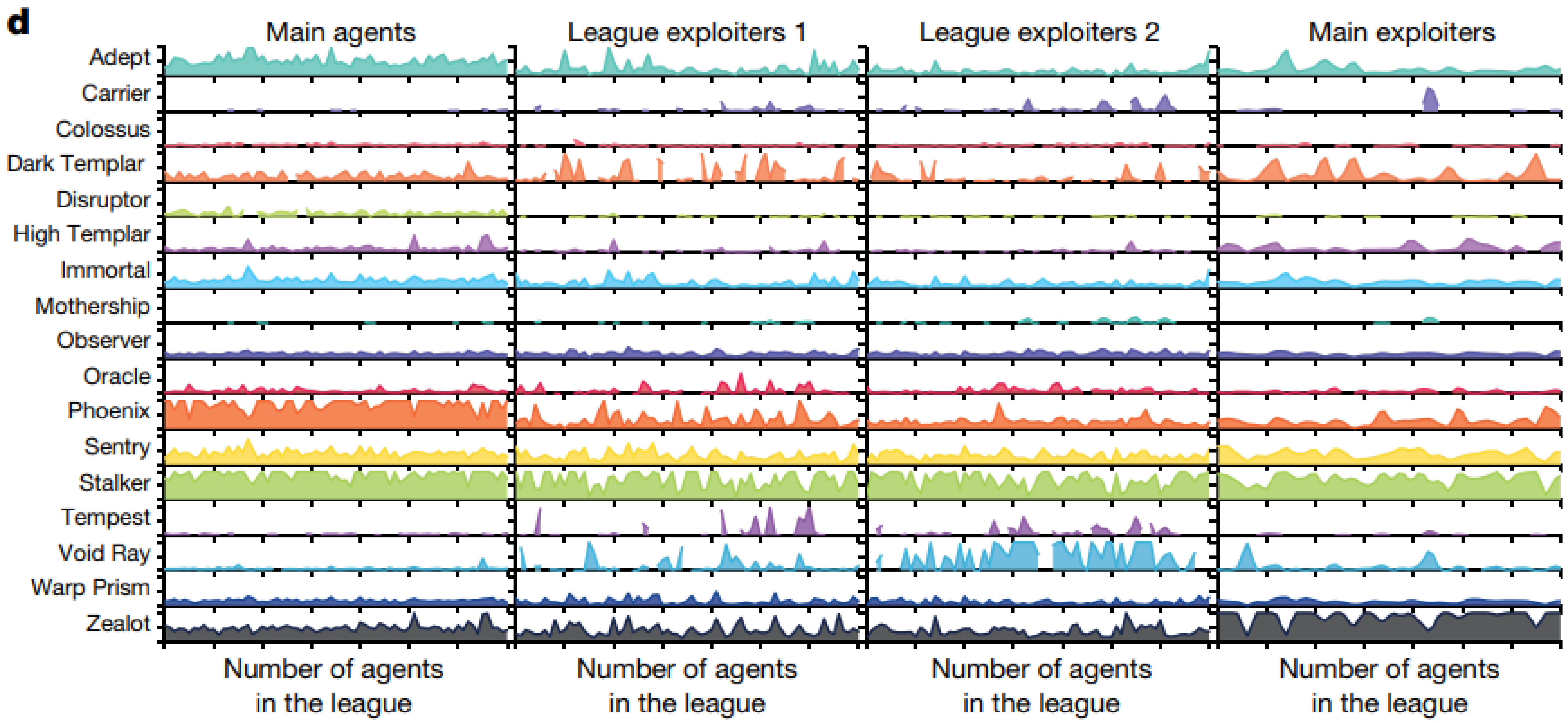


Extended Data Fig. 2 | Delays. Left, distribution of delays between when the game generates an observation and when the game executes the corresponding agent action. Right, distribution of how long agents request to wait without observing between observations.

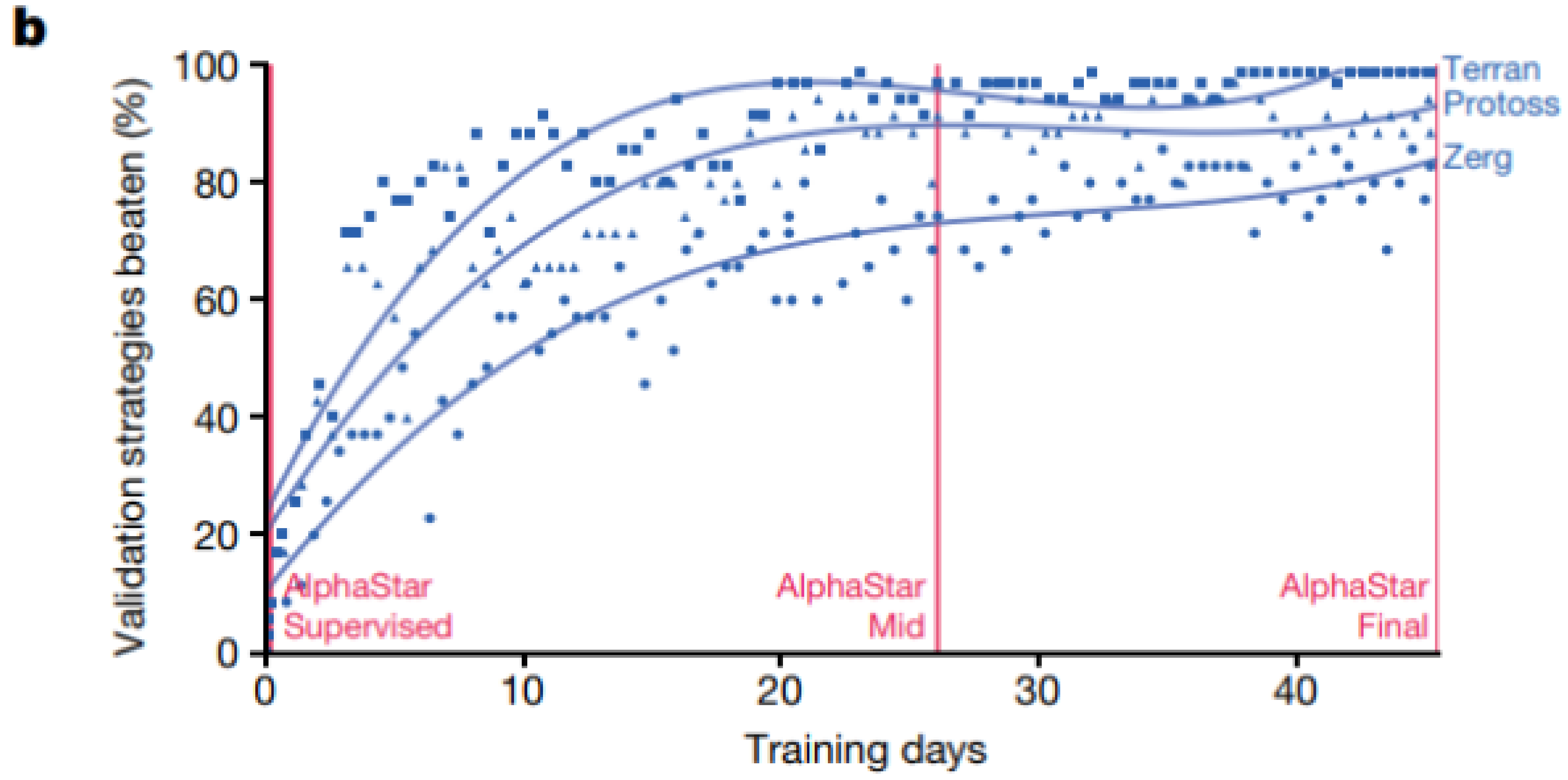
09 Training Results

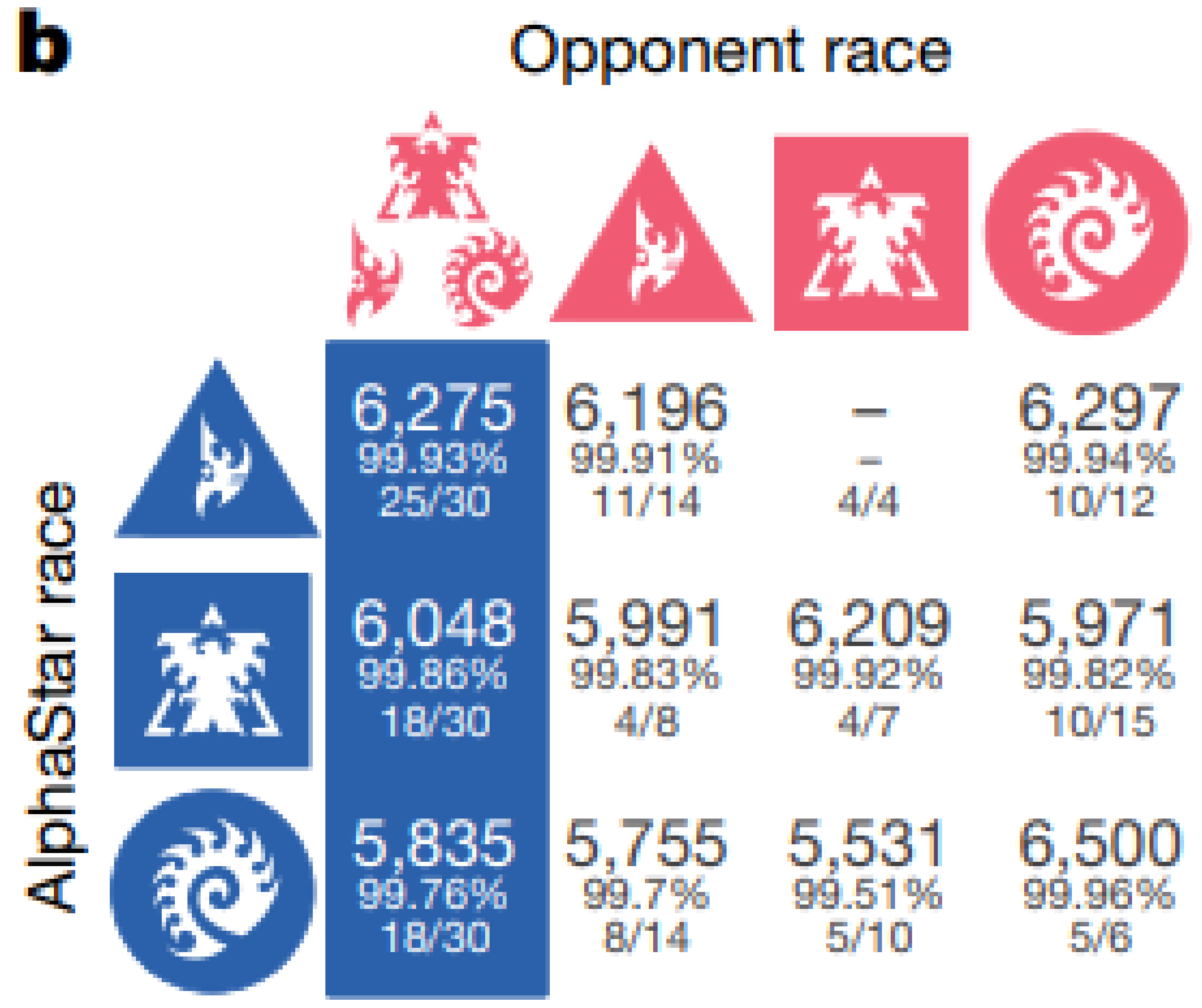


09 Training Results



09 Training Results




b

Opponent race

AlphaStar race

	6,275 99.93% 25/30	6,196 99.91% 11/14	– – 4/4	6,297 99.94% 10/12
	6,048 99.86% 18/30	5,991 99.83% 4/8	6,209 99.92% 4/7	5,971 99.82% 10/15
	5,835 99.76% 18/30	5,755 99.7% 8/14	5,531 99.51% 5/10	6,500 99.96% 5/6

감사합니다

Do you have any questions?