MUNI

AlphaStar

Grandmaster level in StarCraft II using multi-agent reinforcement learning

Vojtěch Formánek
xforman@fi.muni.cz

Masaryk University

December 27, 2025

Today

- Short Intro
- The League
- Agent Structure
- Learning

Population-Based MARL

- Self-play on general-sum games
- Base idea
  1. initialize population
  2. evaluate
  3. modify

Policy Space Response Oracles

Algorithm 26 Policy space response oracles (PSRO)

Initialize populations Π_i^1 for all $i \in I$ (e.g., random policies)
for each generation $k = 1, 2, 3, \ldots$ **do**
  Construct meta-game M^k from current populations $\{\Pi_i^k\}_{i \in I}$
  Use meta-solver on M^k to obtain distributions $\{\delta_i^k\}_{i \in I}$
  for each agent $i \in I$ **do** ▷ Train best-response policies
    for each episode $e = 1, 2, 3, \ldots$ **do**
      Sample policies for other agents $\pi_{-i} \sim \delta_{-i}^k$
      Use single-agent RL to train π_i' wrt. π_{-i} in underlying game G
    Grow population $\Pi_i^{k+1} \leftarrow \Pi_i^k \cup \{\pi_i'\}$

Alphastar

Research problems

- StarCraft II – real-time strategy
- Large state and action space
- Partially observable state space
- Asynchronous training $\rightarrow$ off-policy $\rightarrow$ combating cycles

Input State 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

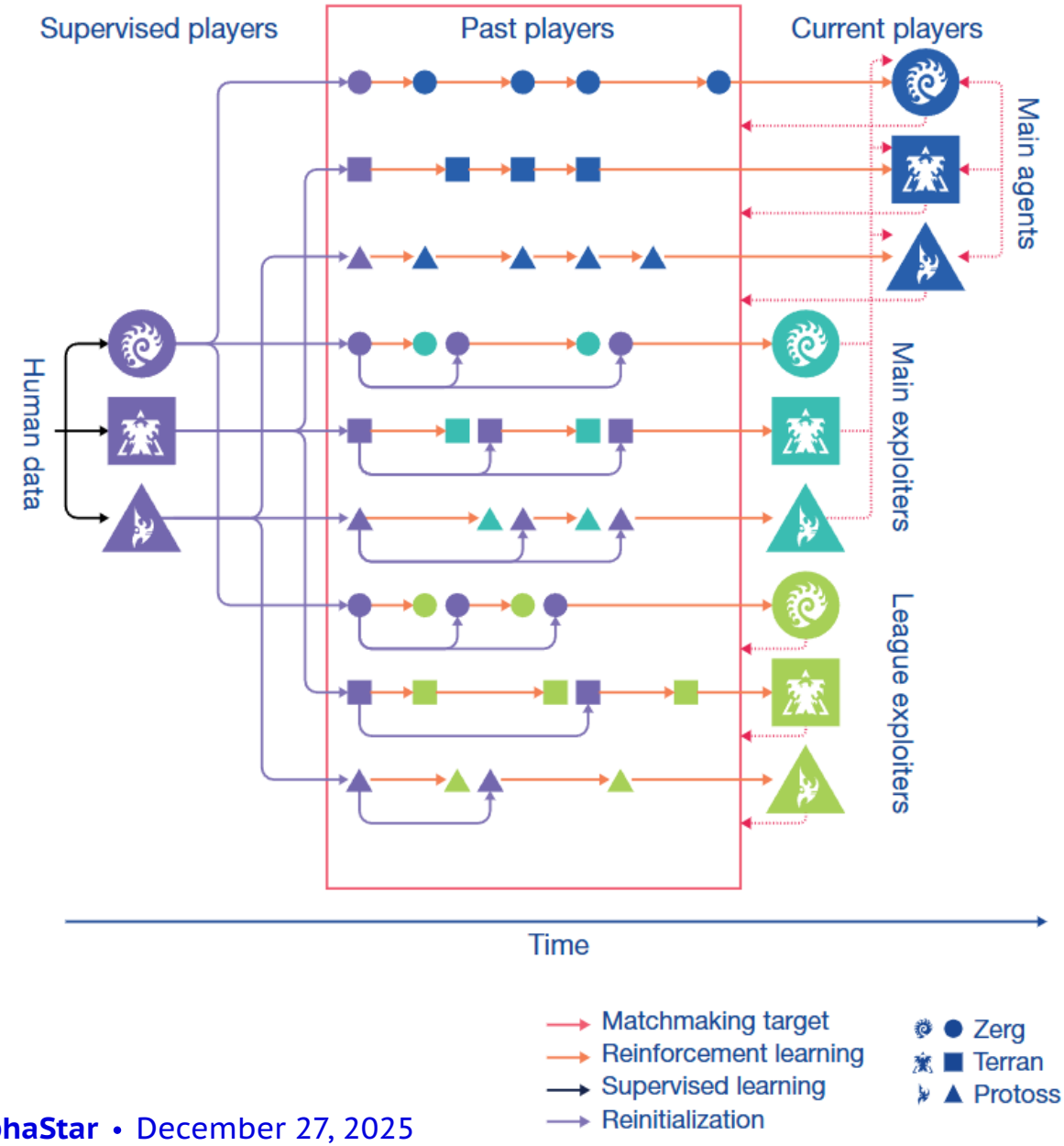
Action Space

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

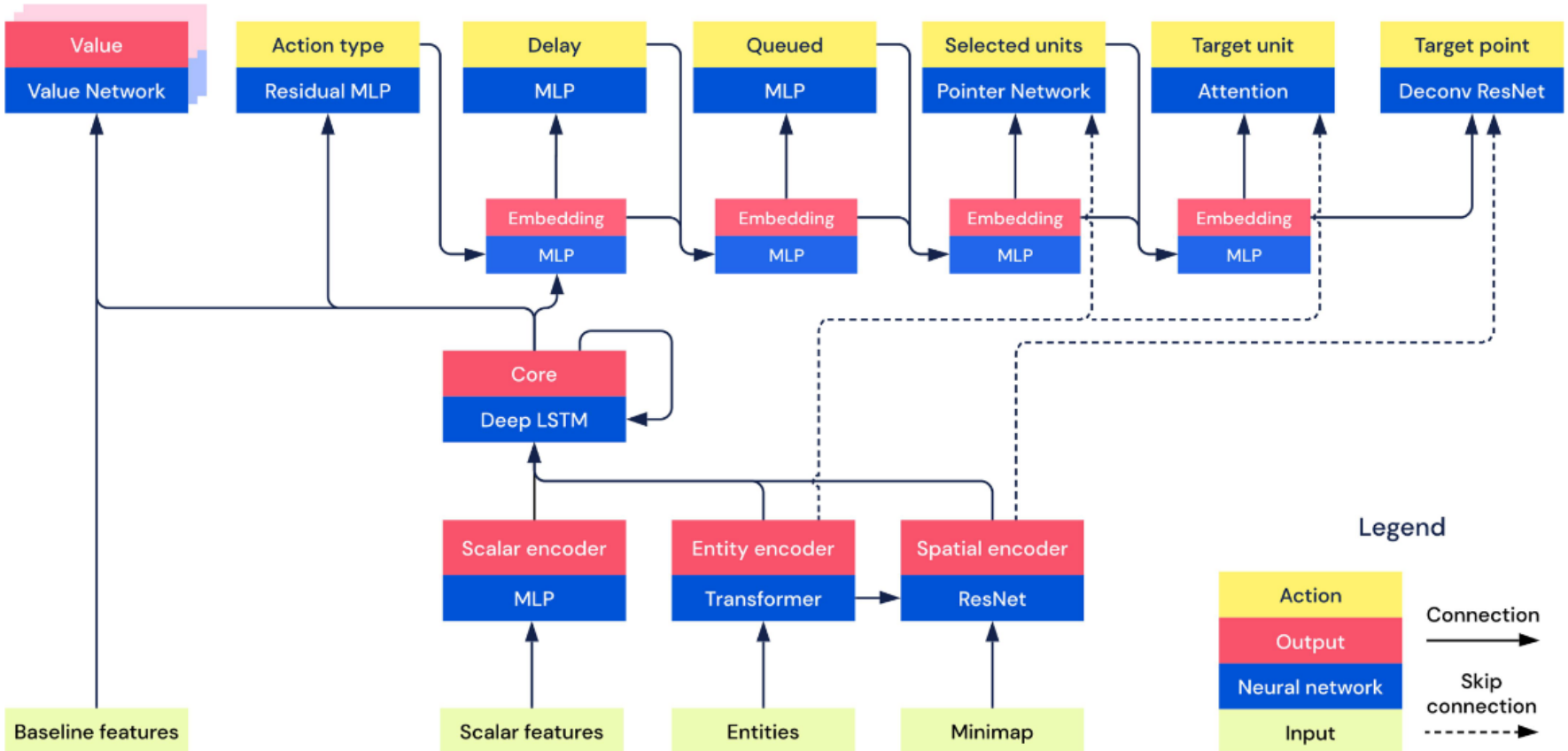
Alphastar

- A PB-MARL algo
- 3 types of agents in the population
  - Main – used for evaluation
  - Main Exploiter – identify exploits in main agents
  - League Exploiter – identify exploits in all agents
- initialized using SL $\rightarrow$ single-agent RL

The Population

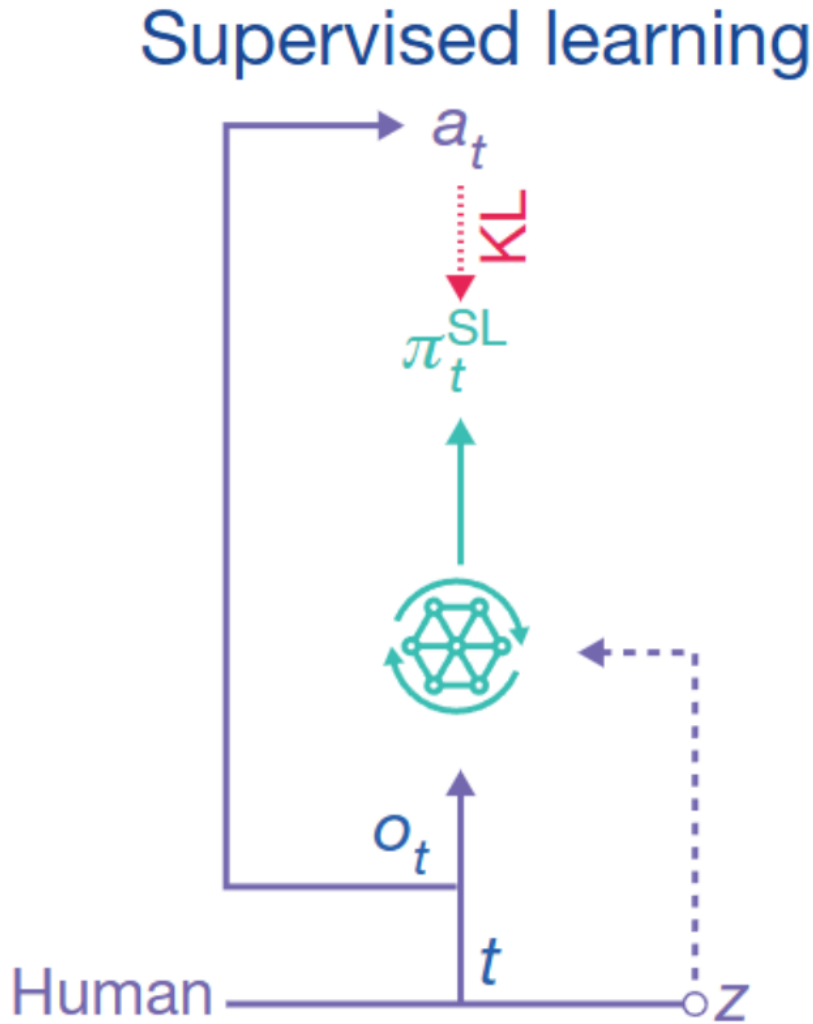


Agent Structure



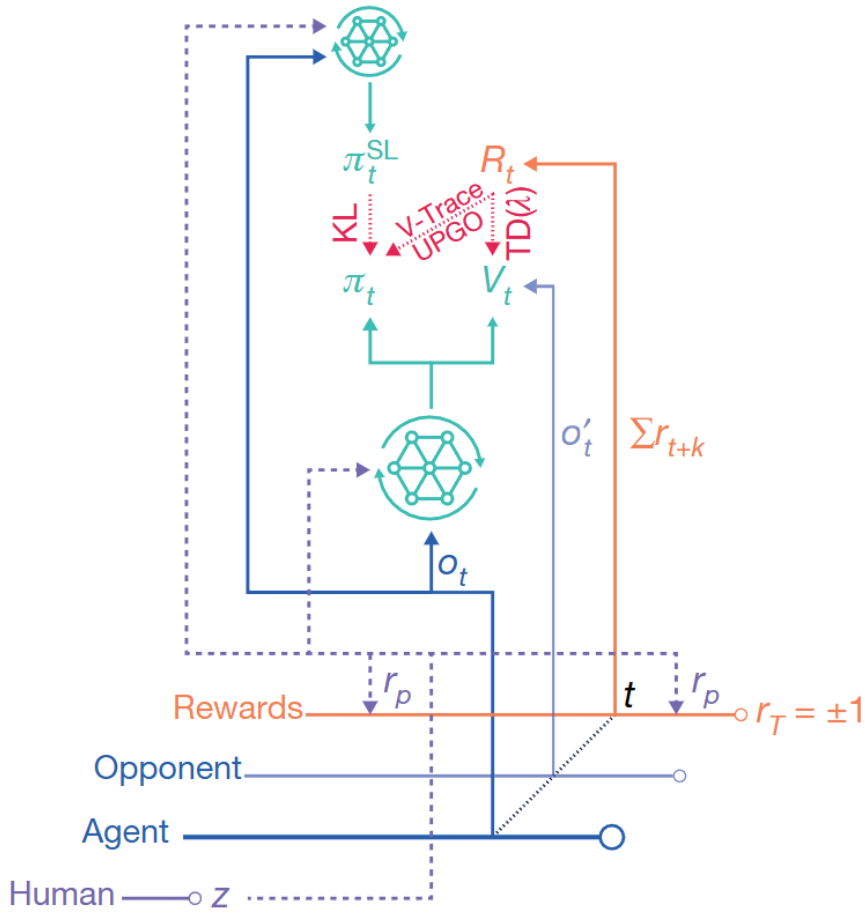
Supervised Learning

- 971K replays
- extract z = build order + cumulative statistics
- D_{KL} diff optimized using Adam ($+L2$ reg.)



Reinforcement Learning

- Policy $\pi_\theta(a_t|s_t, z)$ is an NN
  - $s_t = (o_{1:t}, a_{1:t-1})$
- policy update using V-Trace + UPGO
- Value estimates using $TD(\lambda)$
- $Q(s_t, a_t, z) = r_t + V_\theta(s_{t+1}, z)$



V-trace

DeepMind, 2018

- off-policy correction using truncated IS

Given a trajectory $\tau = s_t, a_t, r_s, \ldots, s_{s+n}, a_{s+n}, r_{s+n}$ *generated by* π*, then V-Trace target for* $V(x_s)$ *is defined as:*

$$v_t = V(s_t) + \sum_{i=s}^{t+n-1} \gamma^{i-s} \left(\prod_{l=s}^{i-1} c_l \right) \rho_t \Big[r_t + \gamma V(s_{t+1}) - V(s_t) \Big],$$

where $\rho_t = min\left(\overline{\rho}, \frac{\pi(a_t|s_t)}{\pi'(a_t|s_t)}\right)$ and $c_t = min\left(\overline{c}, \frac{\pi(a_t s_t)}{\pi'(a_t|s_t)}\right)$

V-trace

DeepMind, 2018

- off-policy correction using truncated IS

Given a trajectory $\tau = s_t, a_t, r_s, \ldots, s_{s+n}, a_{s+n}, r_{s+n}$ *generated by* π*, then V-Trace target for* $V(x_s)$ *is defined as:*

$$v_t = V(s_t) + \sum_{i=s}^{t+n-1} \gamma^{i-s} \left(\prod_{l=s}^{i-1} c_l \right) \rho_t \Big[r_t + \gamma V(s_{t+1}) - V(s_t) \Big],$$

where $\rho_t = min\left(\overline{\rho}, \frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)}\right)$ and $c_t = min\left(\overline{c}, \frac{\pi'(a_t s_t)}{\pi(a_t|s_t)}\right)$

- truncation levels are such that $\overline{\rho} \geq \overline{c}$

V-Trace in AlphaStar

- Updates Policy

$$v_t = V_\theta(s_t) + \sum_{i=s}^{t+n-1} \gamma^{i-s} \left(\prod_{l=s}^{i-1} c_l \right) \rho_t \Big[r_t + \gamma G^{\lambda}_{t:t+n}(\tau) - V_\theta(s_t) \Big],$$

$$\rho_t \nabla_\theta \log \pi_\theta(a_t|s_t)(r_t + \gamma v_{t+1} - V_\theta(s_t)$$

- Problem – truncates too early

V-Trace Problems and Solutions

- Truncates too early
- assume independence between action type, delay and all arguments
- components updated separately
- opponent's observations (z) to decrease variance of V estimates
- UPGO

Upgoing Policy Update (UPGO)

Policy update:

$$\rho_t(G_t^U - V_\theta(S_t, z))\nabla_\theta \log \pi_\theta(a_t|s_t, z),$$

where

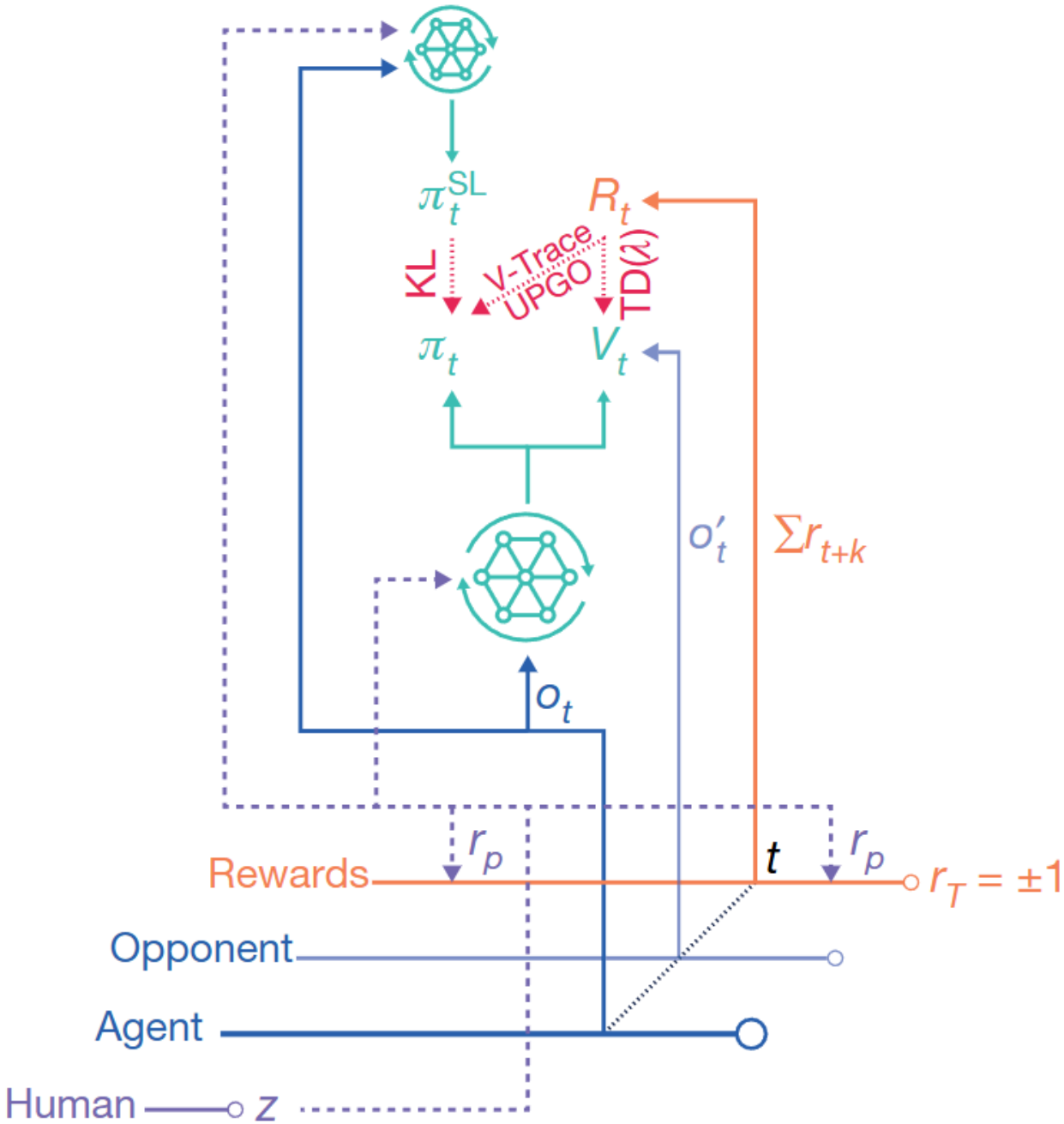
$$G_t^U = \begin{cases} r_t + G_{t+1}^U & \text{if } Q(s_{t+1}, a_{t+1}, z) \geq V_\theta(s_{t+1}, z) \\ r_t + V_\theta(s_{t+1}, z) & \text{otherwise} \end{cases}$$

and $\rho_t = \min\left(1, \frac{\pi'(a_t|s_t)}{\pi(a_t|s_t)}\right)$

Training loss

Weighted sum of:

- Policy loss – V-trace + UPGO
- Value loss – $TD(\lambda)$
- D_{KL} w.r.t the SL policy
- reward r_t + pseudo-rewards based on human data
- entropy regularization loss

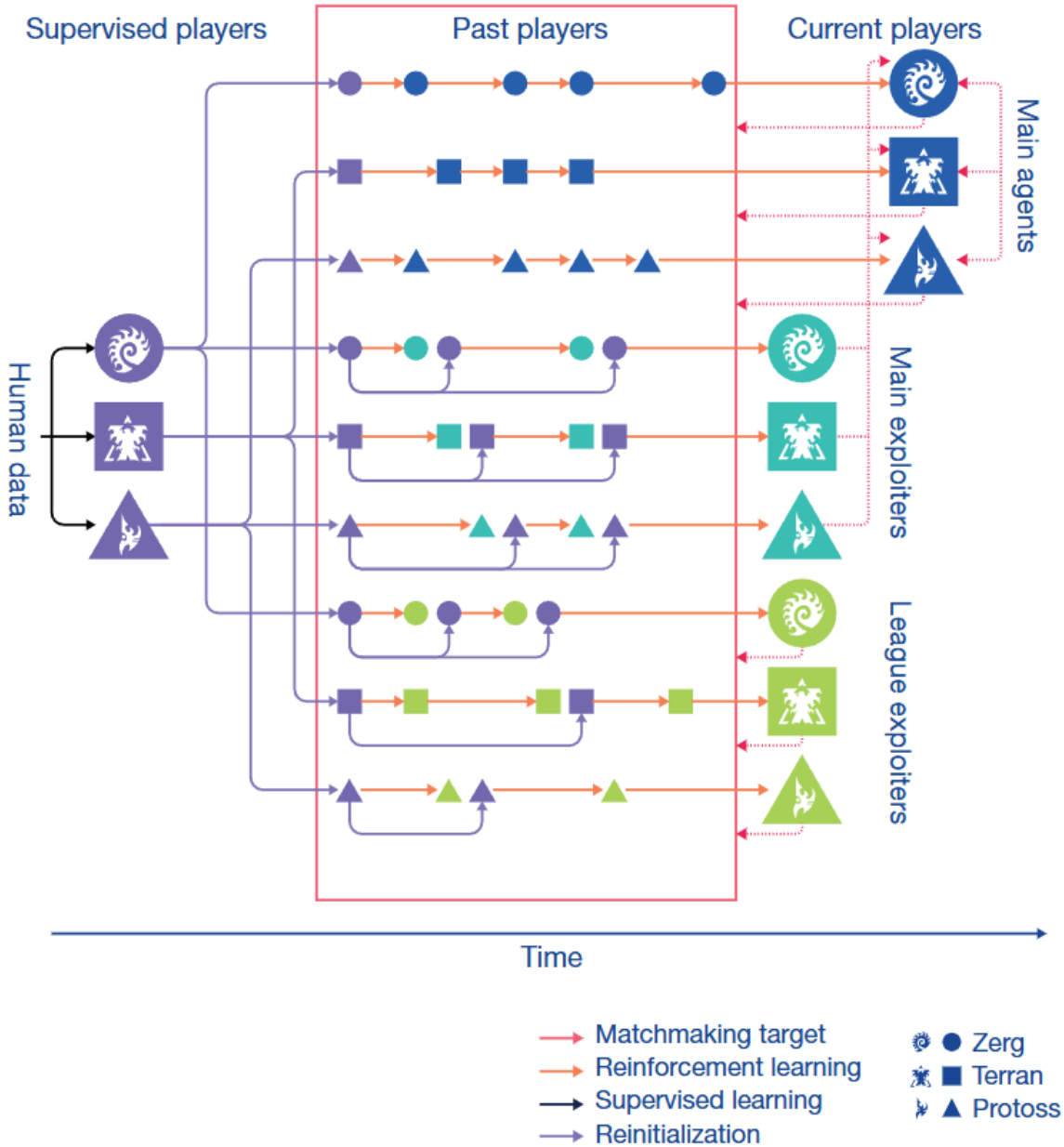

π_t^{SL}
R_t
KL
V-Trace
UPGO
TD(λ)
π_t
V_t
o'_t
Σr_{t+k}
o_t
r_p
t
r_p
Rewards
$r_T = \pm 1$
Opponent
Agent
Human
z

Describing the League

- Main purpose – combating cycles
- uses Prioritized Fictitious Self-Play

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

- three types of agents – Main, Main Exploiters, League Exploiters


Supervised players
Past players
Current players
Main agents
Human data
Main exploiters
League exploiters
Time
Matchmaking target
Reinforcement learning
Supervised learning
Reinitialization
Zerg
Terran
Protoss

Agent Opponents

- Main
  - 35% self-play, 50% PFSP league, 15% no-longer beaten main
  - every 2×10^9 steps copy added to league
- League Exploiters – Identify weakness of all agents
  - 100% all Main agents
  - added to league if wrt. > 70% against each agent or timeout
  - 25% reset to SL initialization
- Main Exploiters – identify weakness of main agents
  - 50% main or if wrt < 20% PSFP past main
  - added if all three main beaten > 70%, timeout
  - always reset

Empirical Results