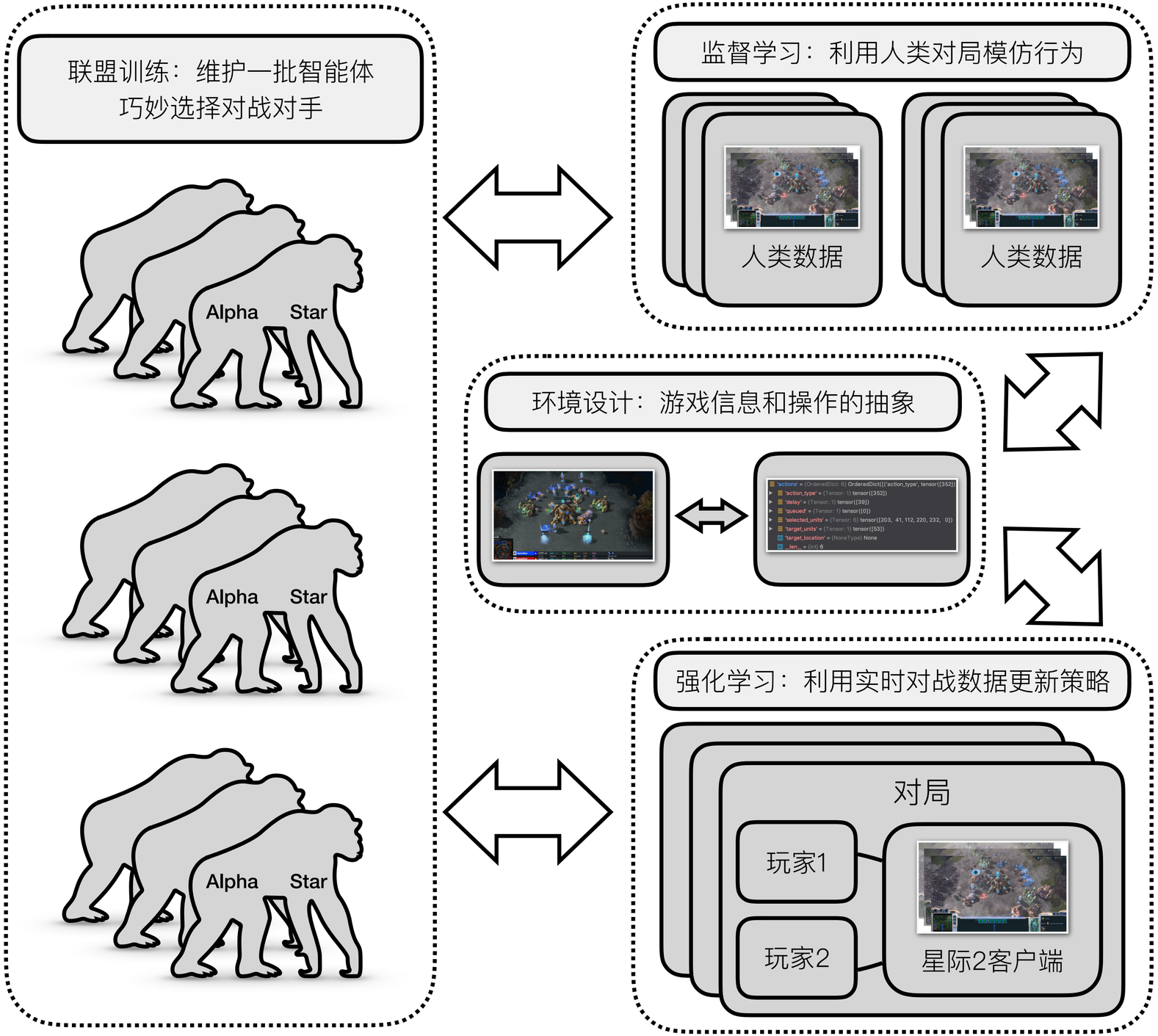


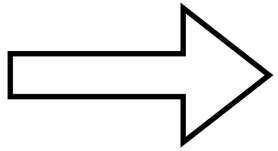
【学术向】AlphaStar浅谈

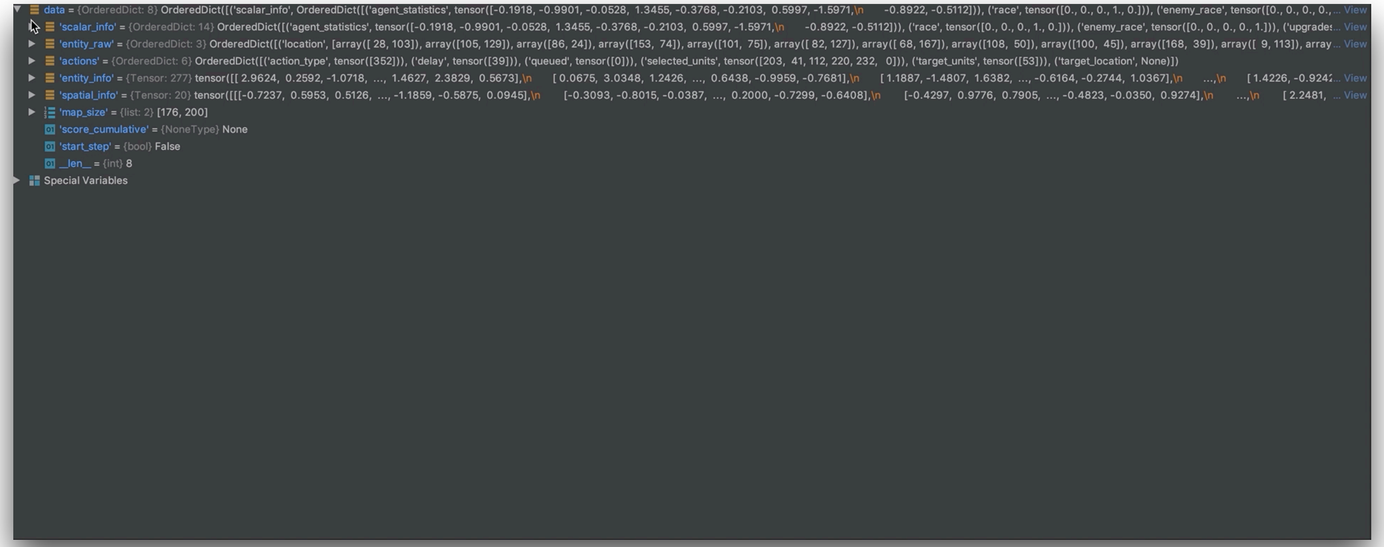
彭正皓

2020年7月

AlphaStar环境设计

状态(观察)是什么





AlphaStar环境设计

状态(观察)是什么

实体

向量

地图

图像

玩家数据

游戏统计

标量数据

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

AlphaStar环境设计

状态(观察)是什么

实体

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

地

玩家

游戏

实体信息

在每个时刻，环境给神经网络N个长度为K的向量，各表示此刻智能体能够看见的N个实体的信息

AlphaStar环境设计

状态(观察)是什么

地图信息

在每个时刻，环境给神经网络20个176乘200的矩阵，各表示此刻全局地图中的一些信息

实体

地图

玩家数据

游戏统计

	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

AlphaStar环境设计

状态(观察)是什么

实体

地图

玩家数据

游戏统计

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

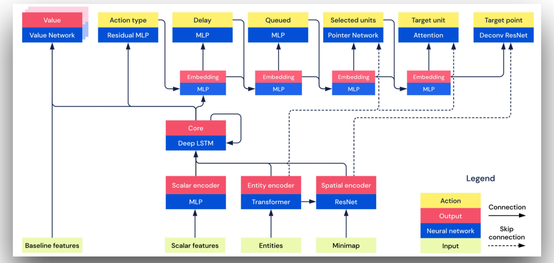
AlphaStar环境设计

动作是什么

观察

神经网络

动作



AlphaStar环境设计

动作是什么

动作类型
（移动单位、升级建筑、移动摄像机）

选中单位

目标
（目的地、攻击对象）

是否立即执行动作

是否重复动作

等候多久才接收
下一次输入

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环境设计

动作是什么

动作类型
（移动单位、升级建筑、移动摄像机）

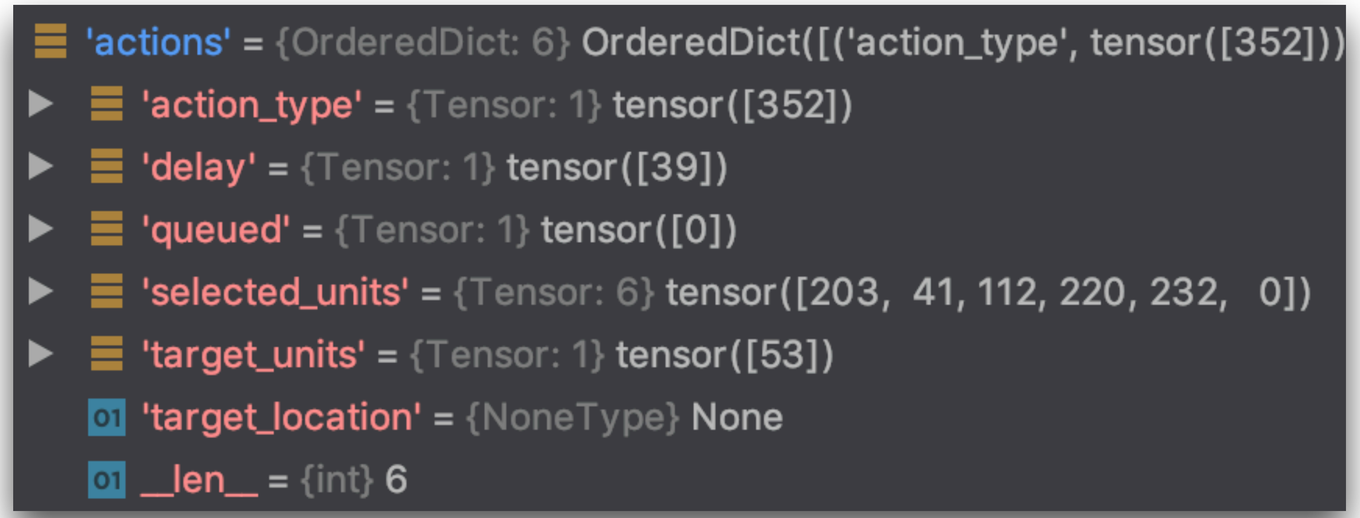
选中单位

目标
（目的地、攻击对象）

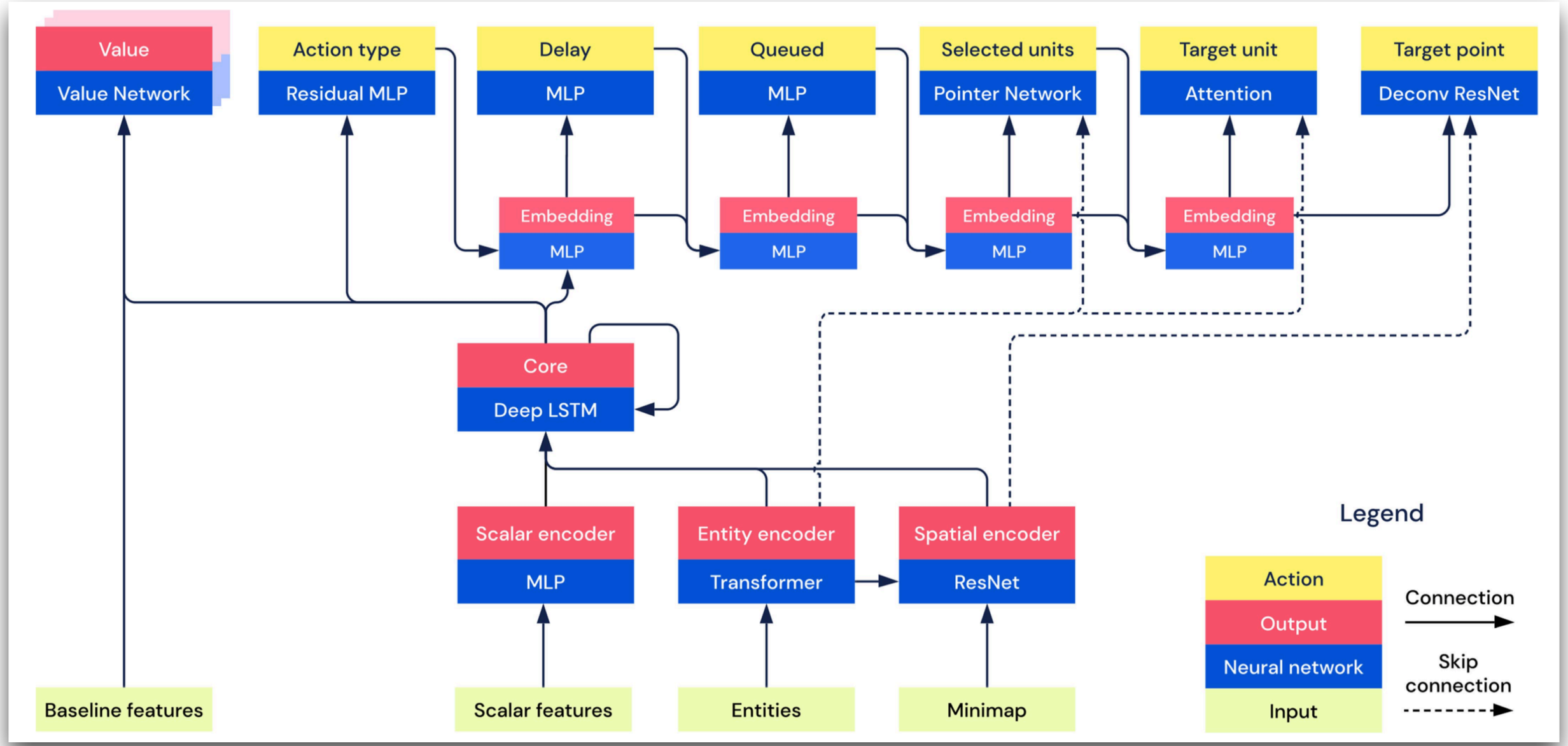
是否立即执行动作

是否重复动作

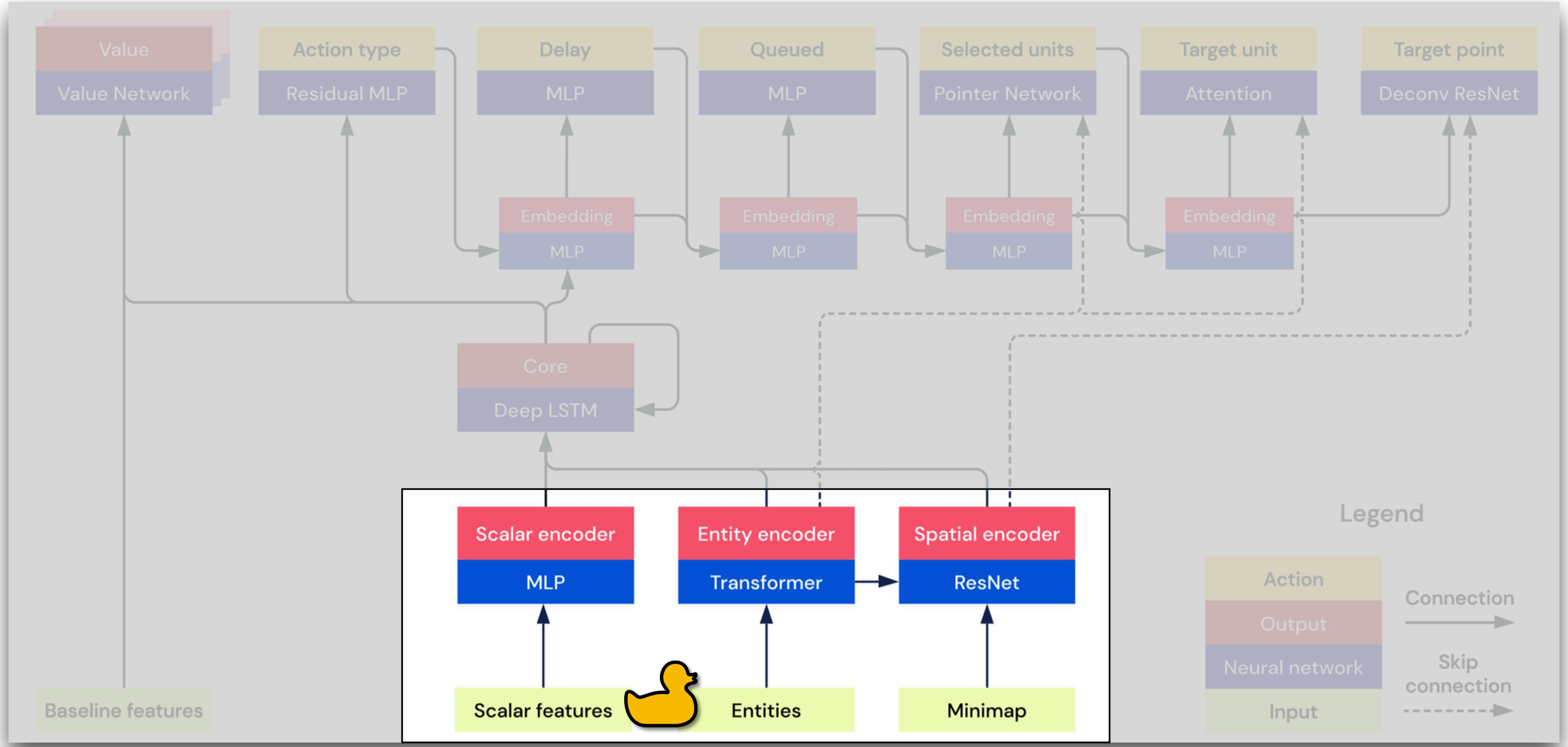
等候多久才接收
下一次输入



AlphaStar网络结构



AlphaStar网络结构



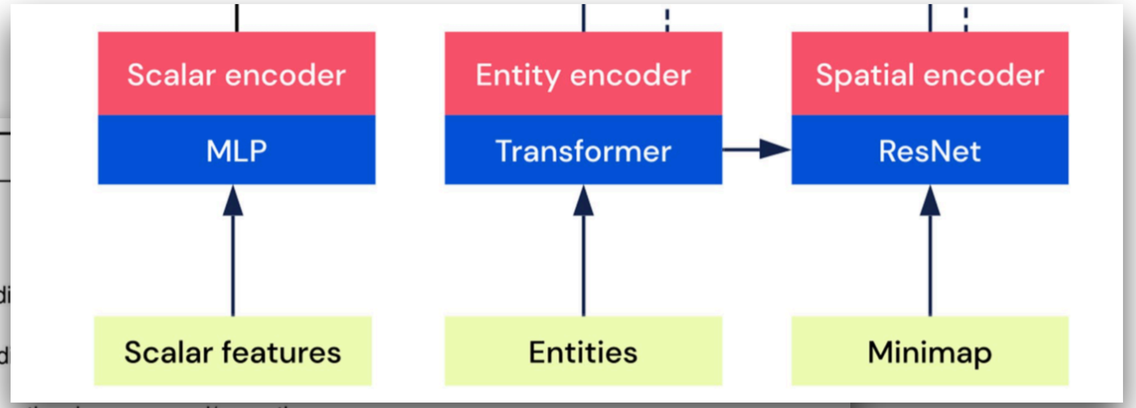
AlphaStar网络结构

实体

地图

玩家数据

游戏统计



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 buildi
	Position	Entity position
	Number of workers	For resource collecting base build
	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

AlphaStar网络结构

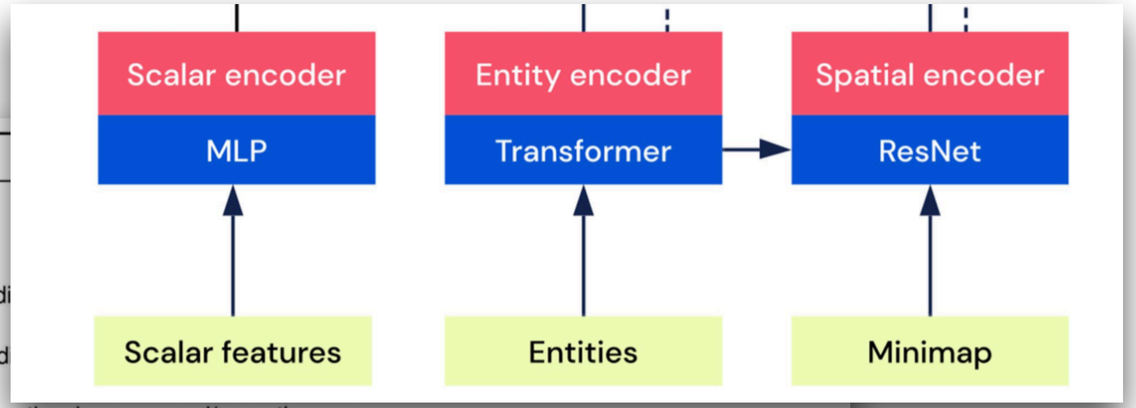
实体

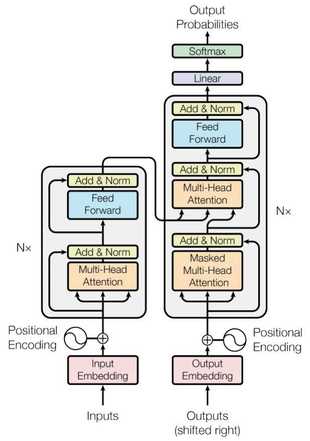
地

玩家

游戏

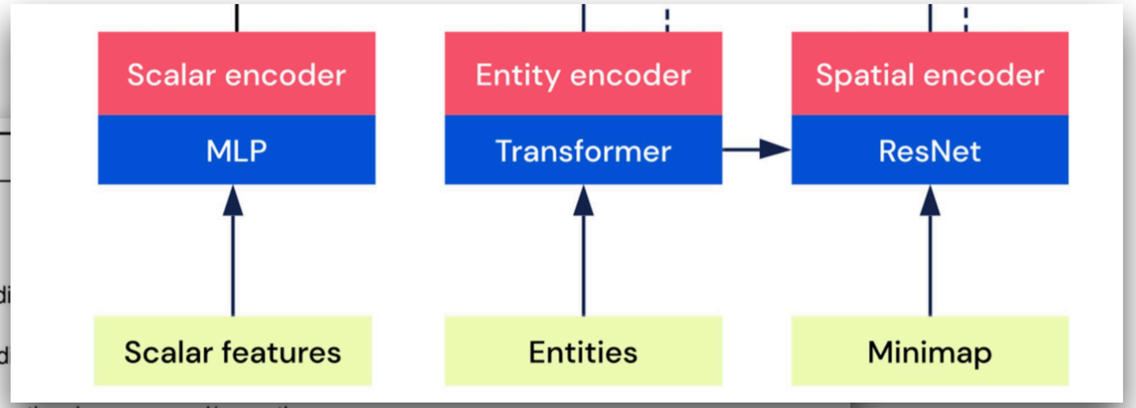
Category	Field	Description
Entities: up to 512	Unit type	E.g. Drone or Forcefield
	Owner	Agent, opponent, or neutral
	Status	Current health, shields, energy
	Display type	E.g. Snapshot, for opponent buildi
	Position	Entity position
	Number of workers	For resource collecting base build
	Cooldowns	Attack 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





每一个实体的信息经Transformer处理变为一个等长的向量，对于所有向量取平均得到Entity embedding

AlphaStar网络结构



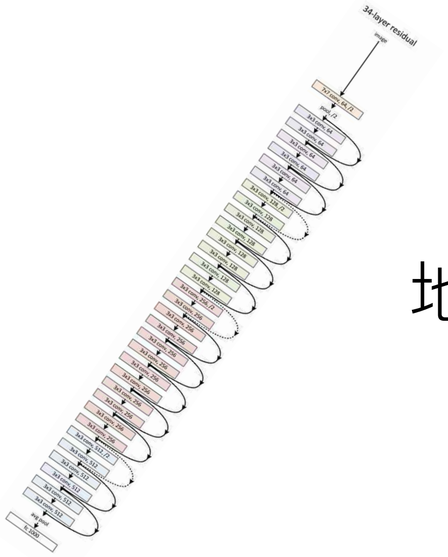
实体

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 buildi
	Position	Entity position
	Number of workers	For resource collecting base build
	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

地

玩家

游戏



地图信息经ResNet处理变为一个定长的向量

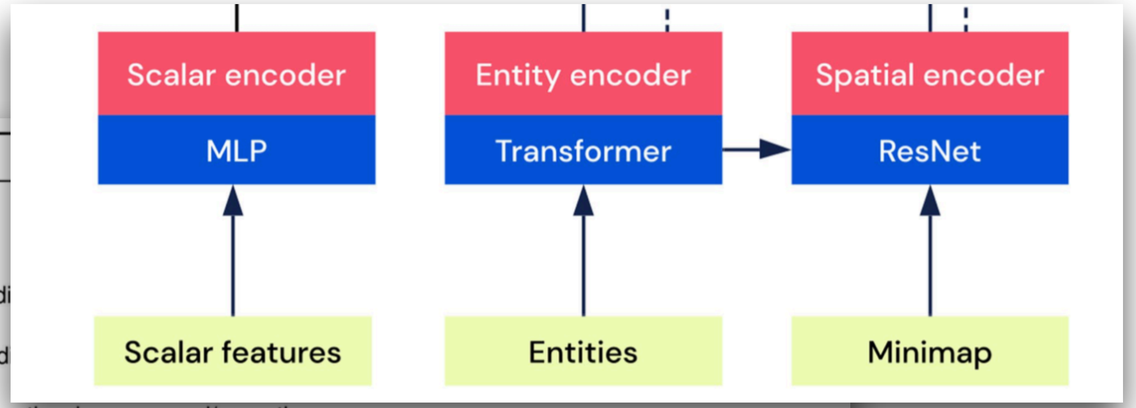
AlphaStar网络结构

实体

地图

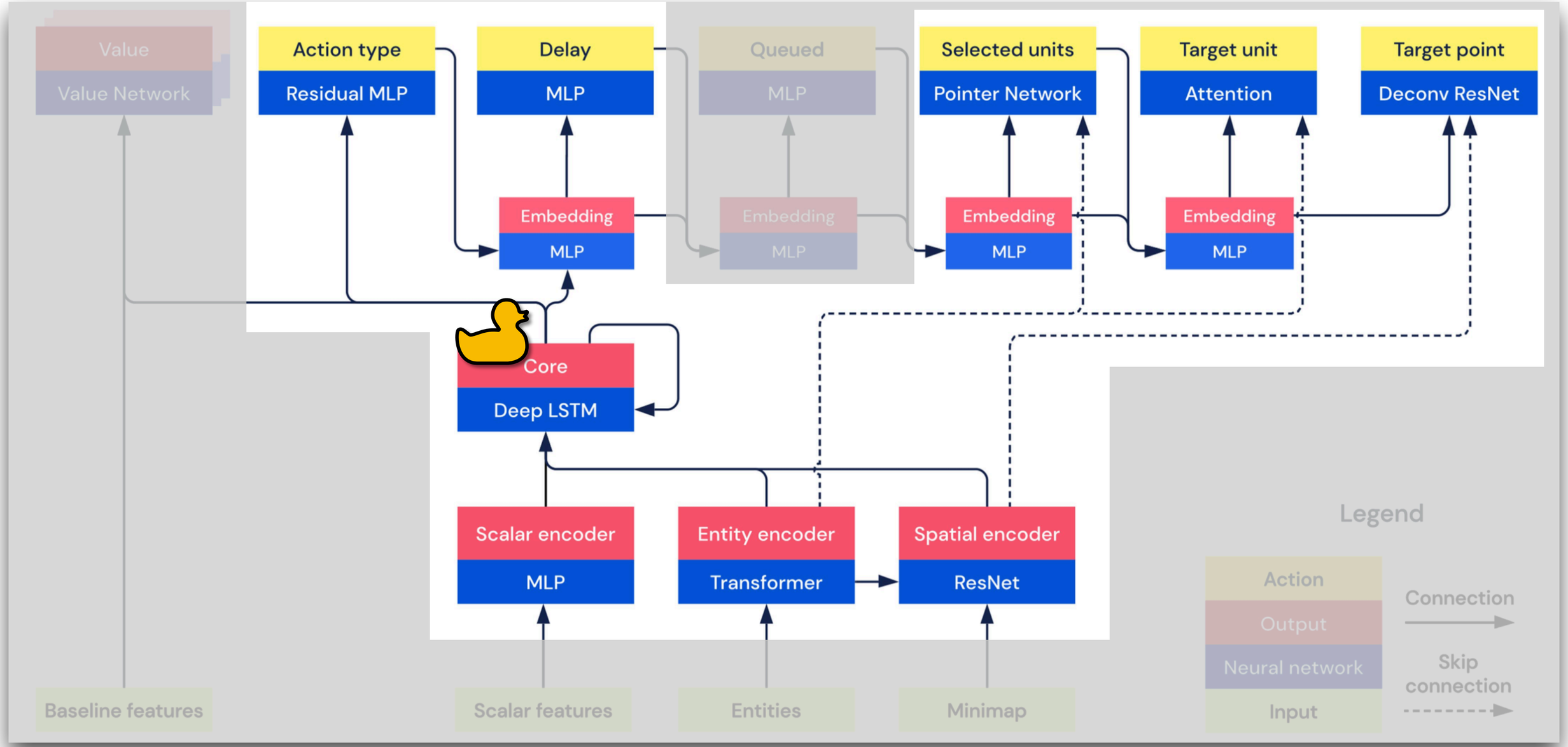
玩家数据

游戏统计

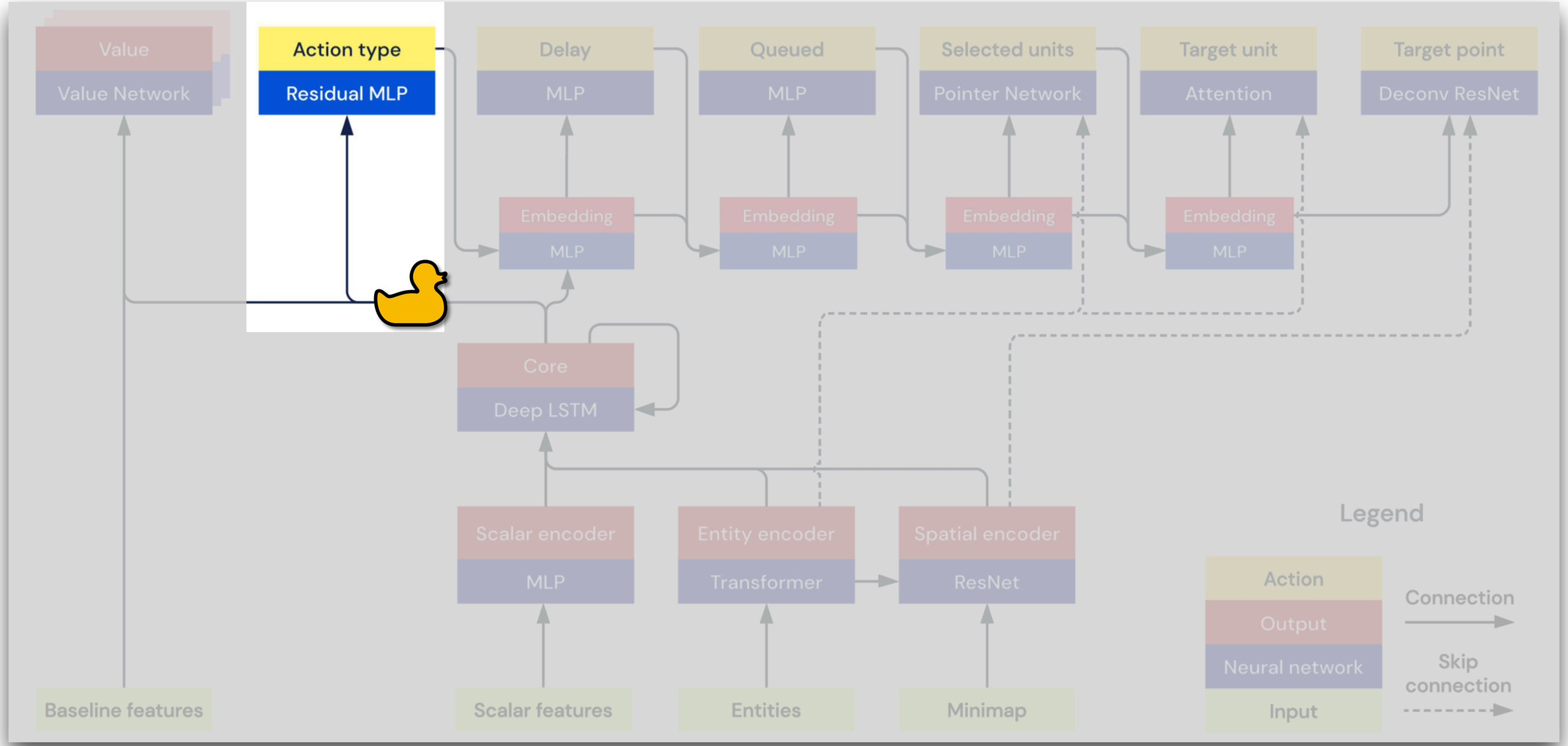


Category	Field	Description
Entities: up to 512	Unit type	E.g. Drone or Forcefield
	Owner	Agent, opponent, or neutral
	Status	Current health, shields, energy
	Display type	E.g. Snapshot, for opponent buildi
	Position	Entity position
	Number of workers	For resource collecting base build
	Cooldowns	Attack 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

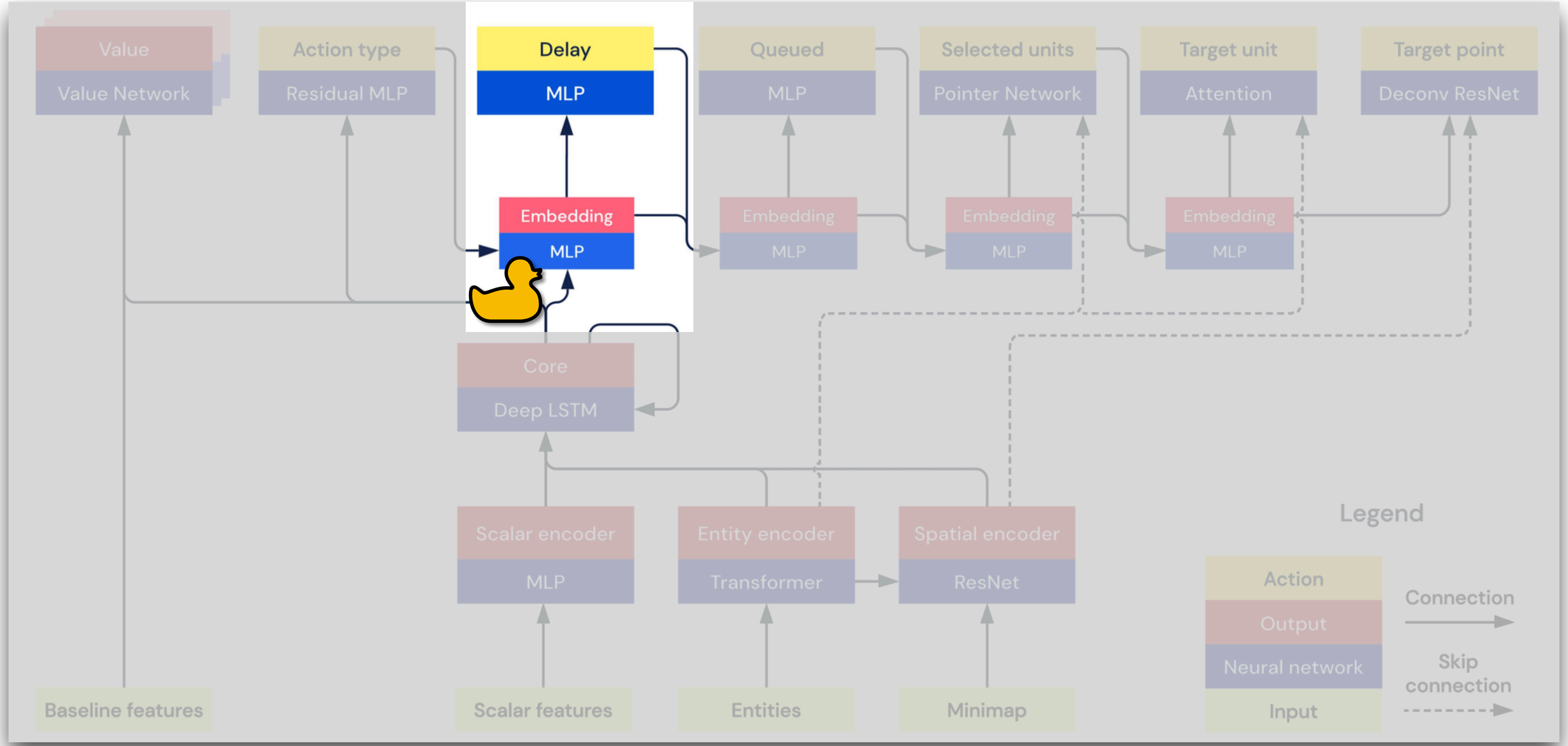
AlphaStar网络结构



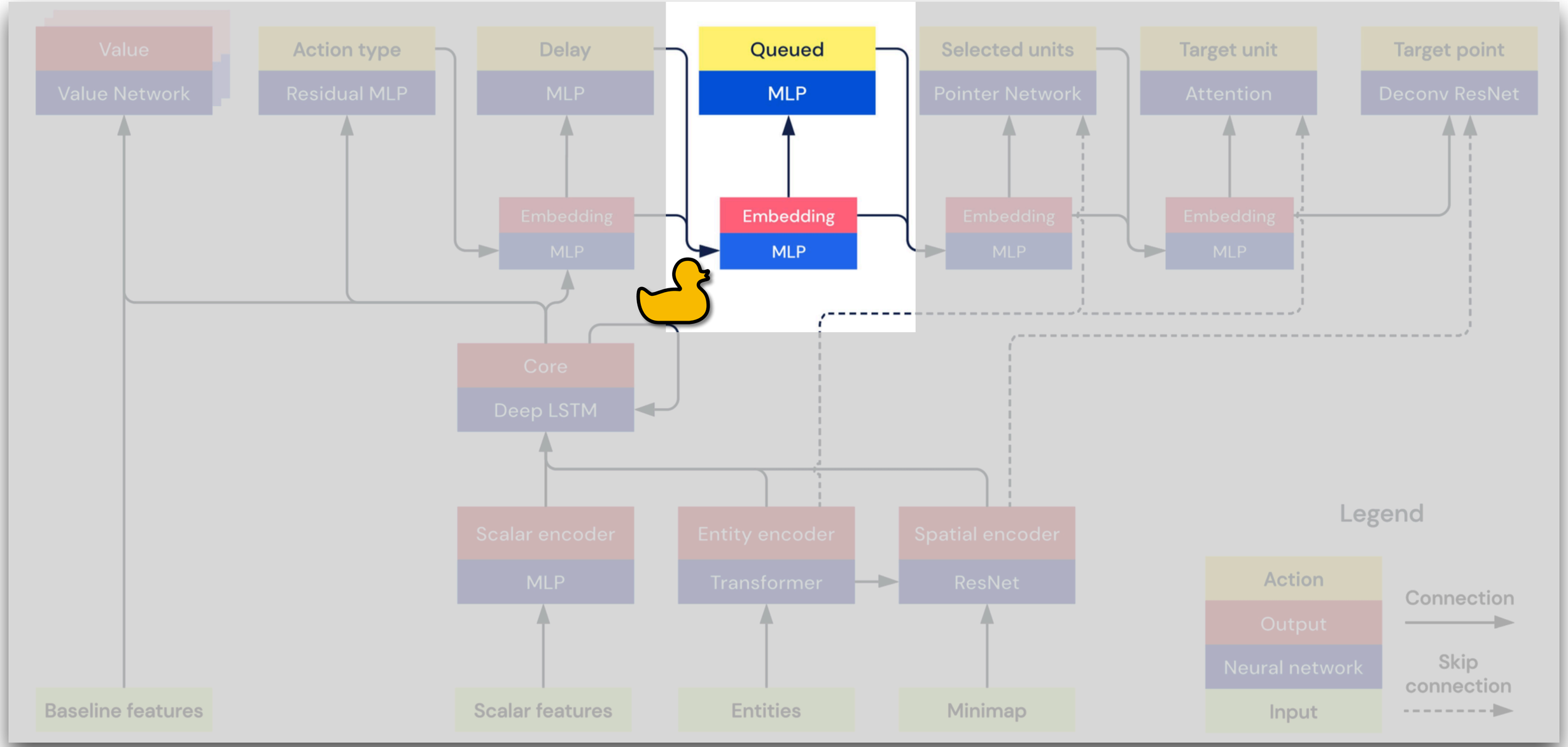
AlphaStar网络结构



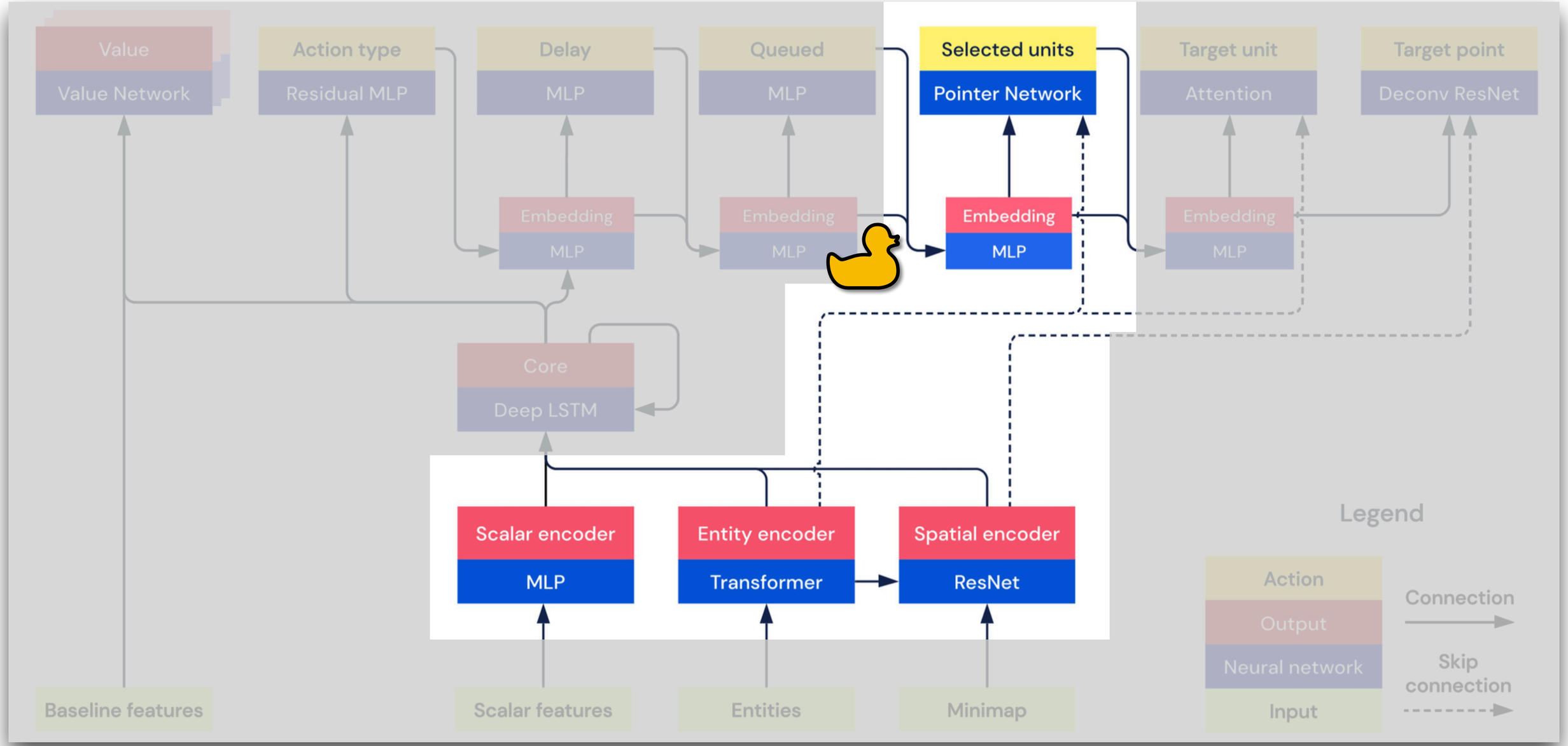
AlphaStar网络结构



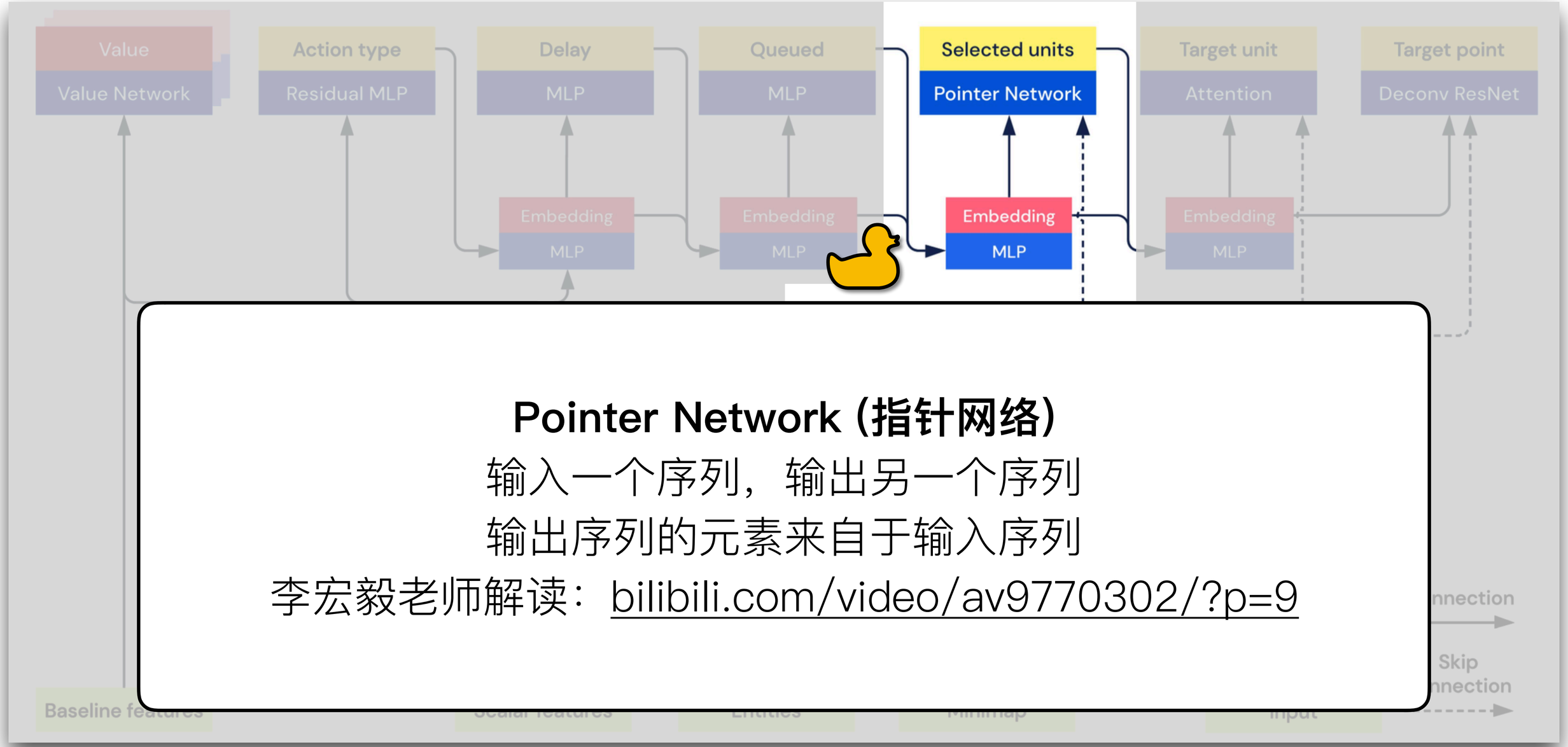
AlphaStar网络结构



AlphaStar网络结构



AlphaStar网络结构



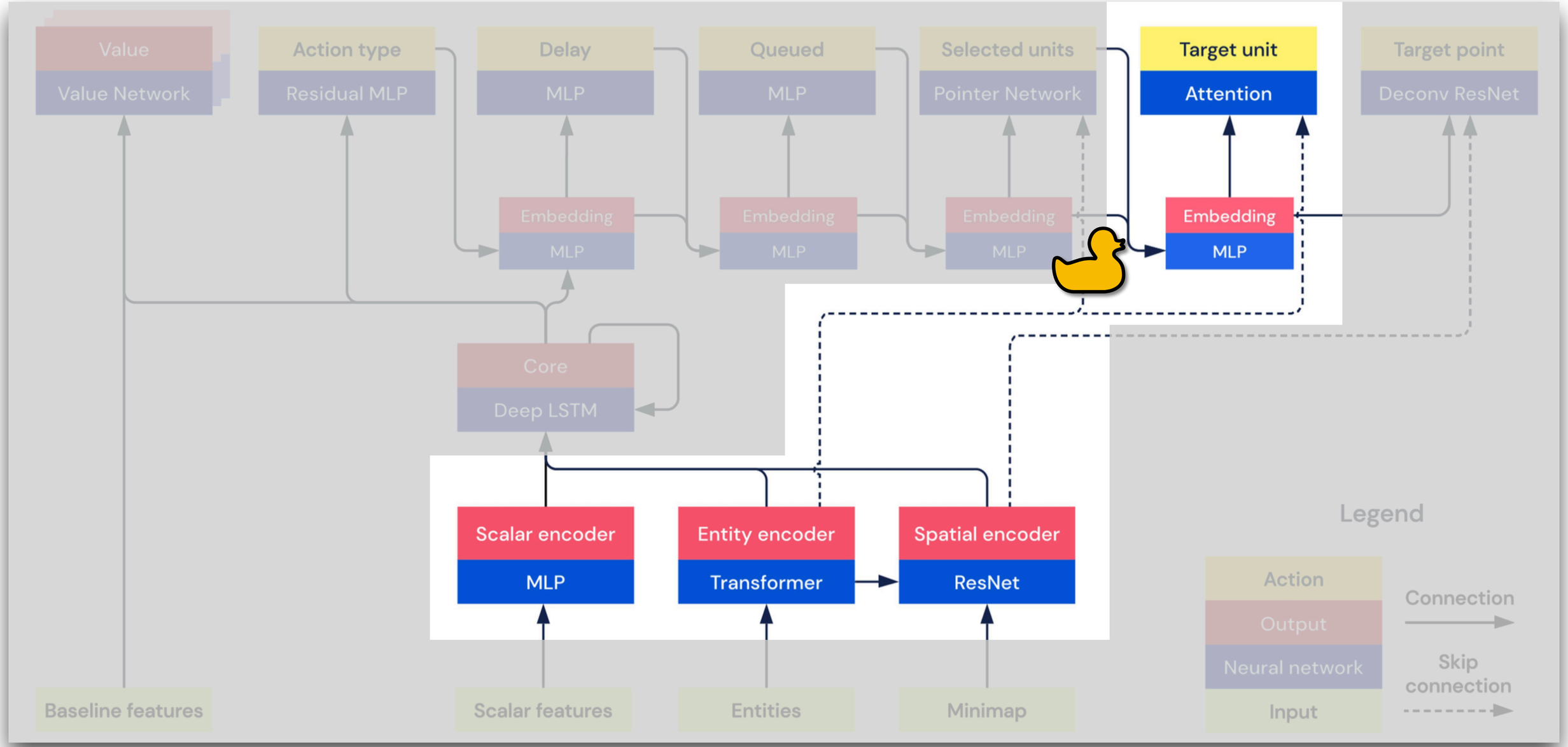
Pointer Network (指针网络)

输入一个序列，输出另一个序列

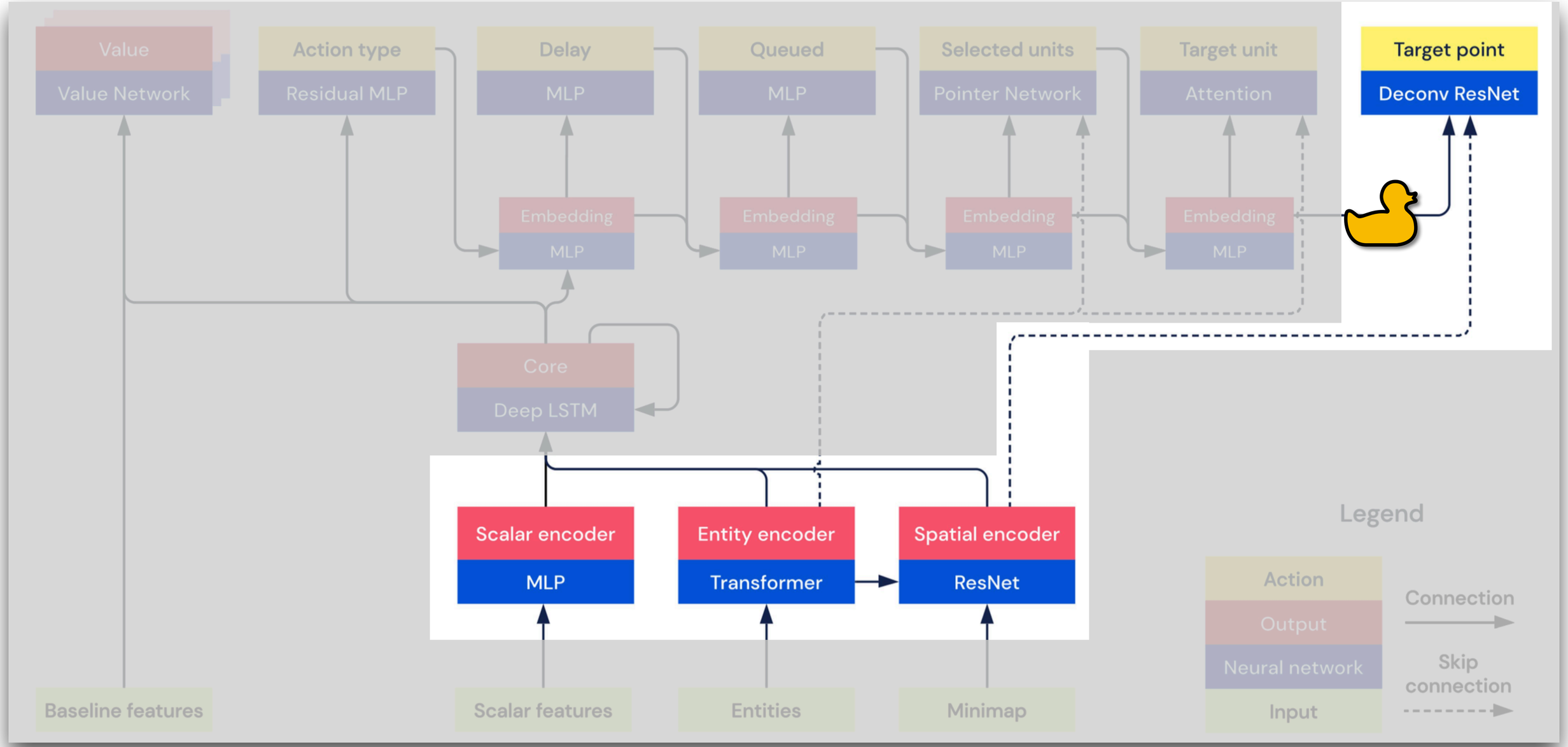
输出序列的元素来自于输入序列

李宏毅老师解读：bilibili.com/video/av9770302/?p=9

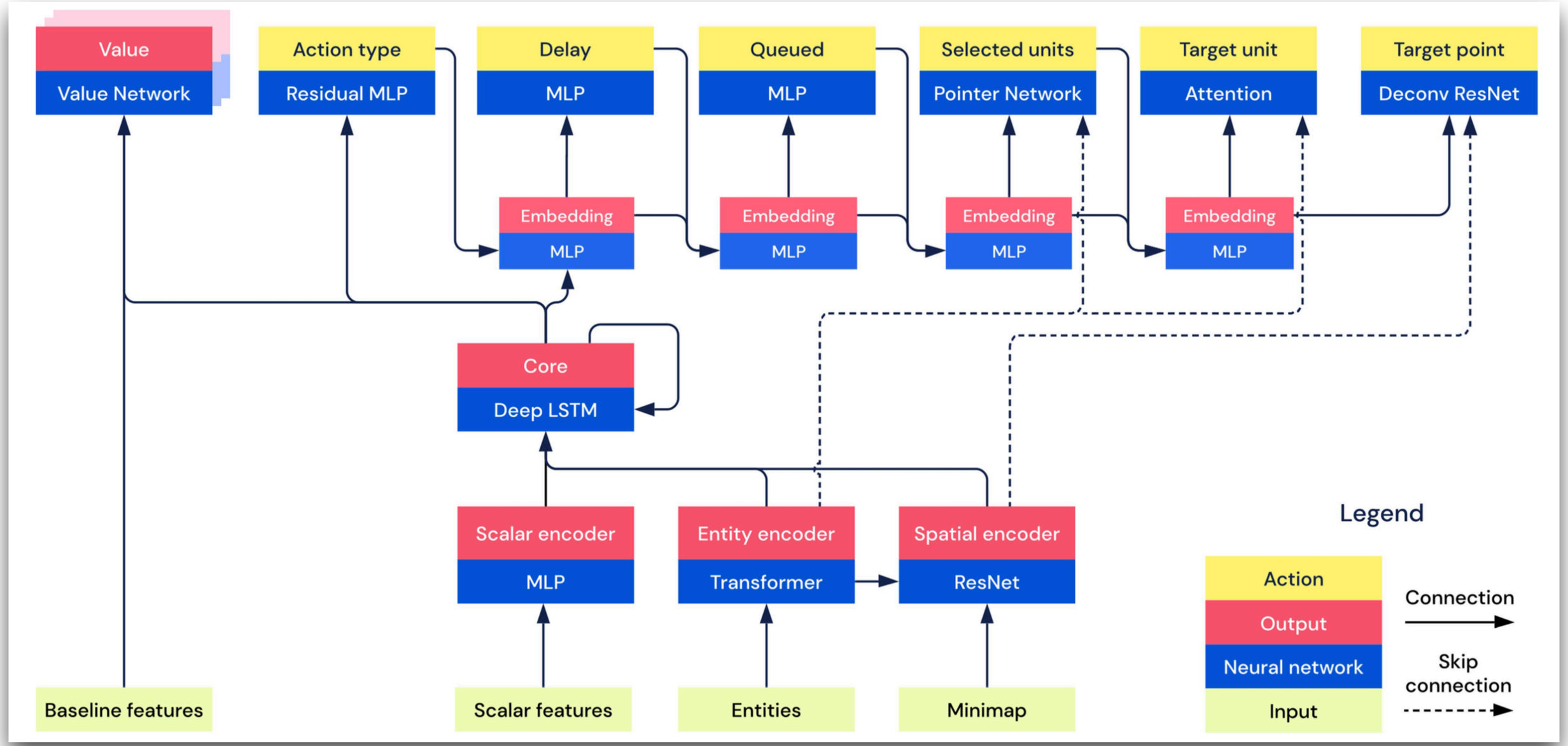
AlphaStar网络结构



AlphaStar网络结构



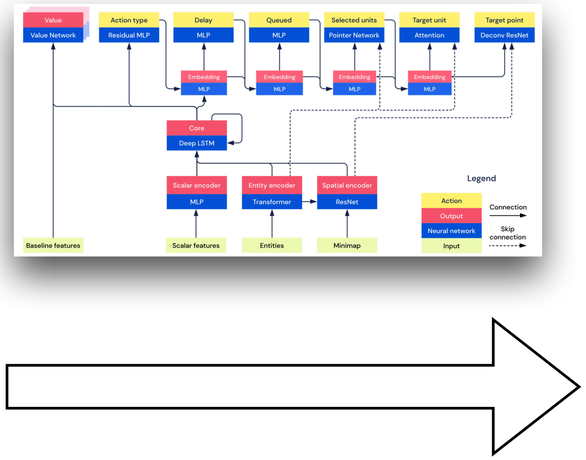
AlphaStar网络结构



AlphaStar网络结构

观察

神经网络



动作

AlphaStar学习算法

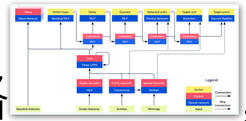
概述

1. 有监督学习：解决神经网络初始化的问题
2. 强化学习：从奖励中学习
3. 模仿学习：配合强化学习，学会模仿人类
4. 多智能体学习(Multi-agent Learning)和自学习(Self-play)：解决如何挑选对手的问题

AlphaStar学习算法

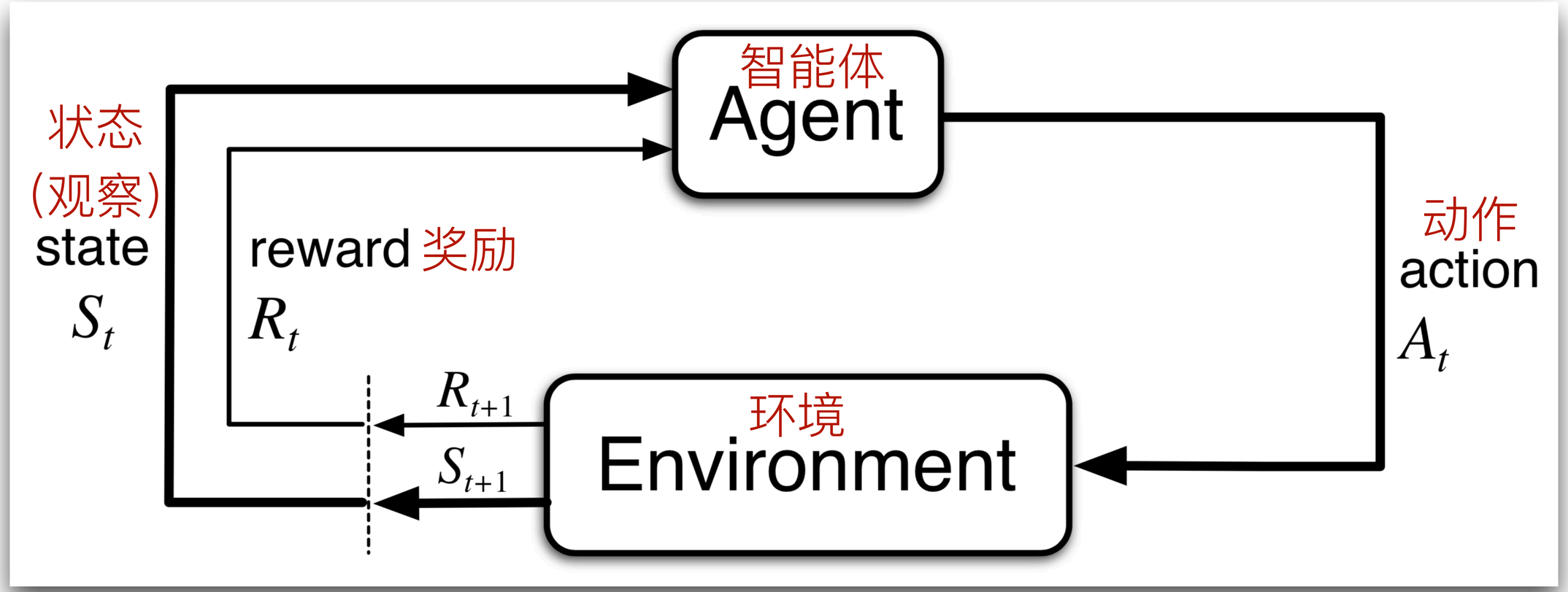
监督学习

- 目的：利用大量的人类数据学会一个比较好的初始化
- 做法：
  1. 解码人类对局，在每个对局中的时刻，解码游戏的状态并得到观察 

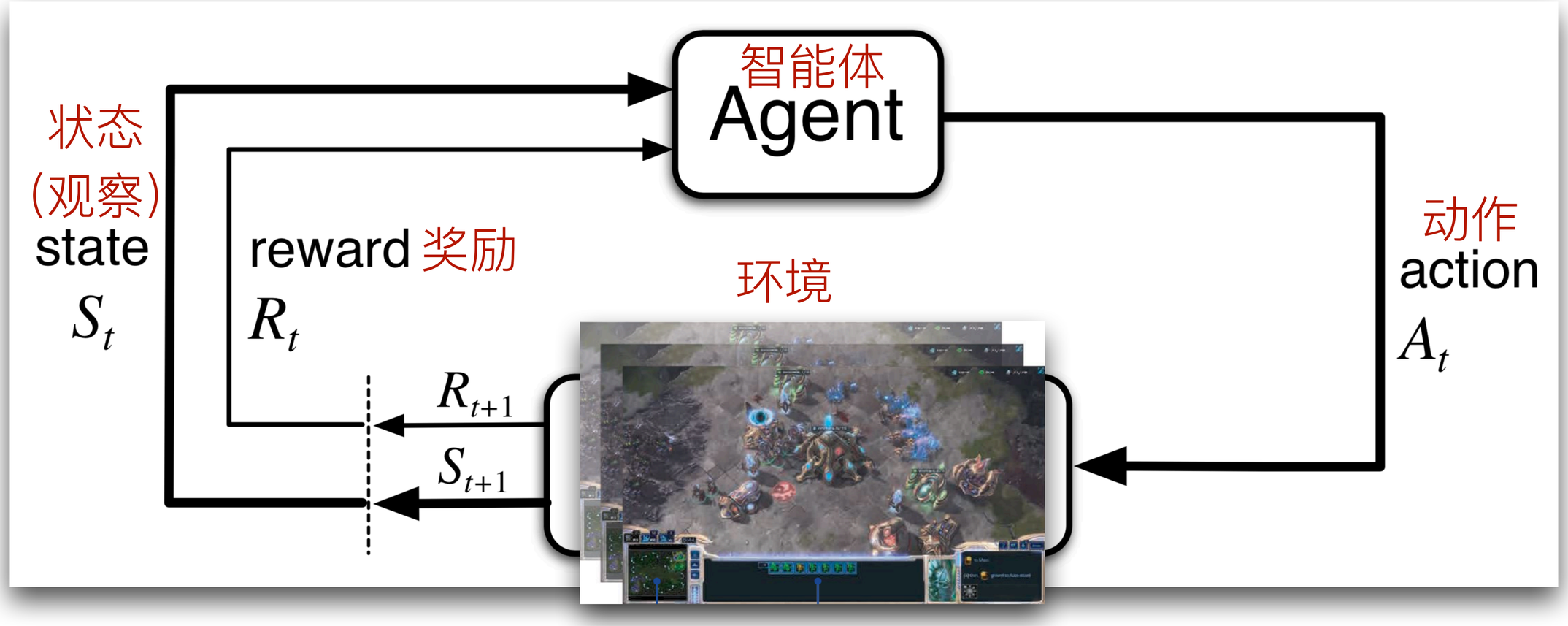
  2. 将观察输入网络  ，得到输出的动作的概率分布 

  3. 计算智能体输出与人类数据的KL loss，并以此优化网络
- 两个细节：
  1. 对于每个head分别进行计算KL loss和优化的步骤，如action type则计算Cross Entropy，target location则计算MSE
  2. 假设这个head的前面的head已经采取了人类的选择

强化学习要素


智能体
Agent
状态
(观察)
state
S_t
reward 奖励
R_t
动作
action
A_t
R_{t+1}
S_{t+1}
环境
Environment

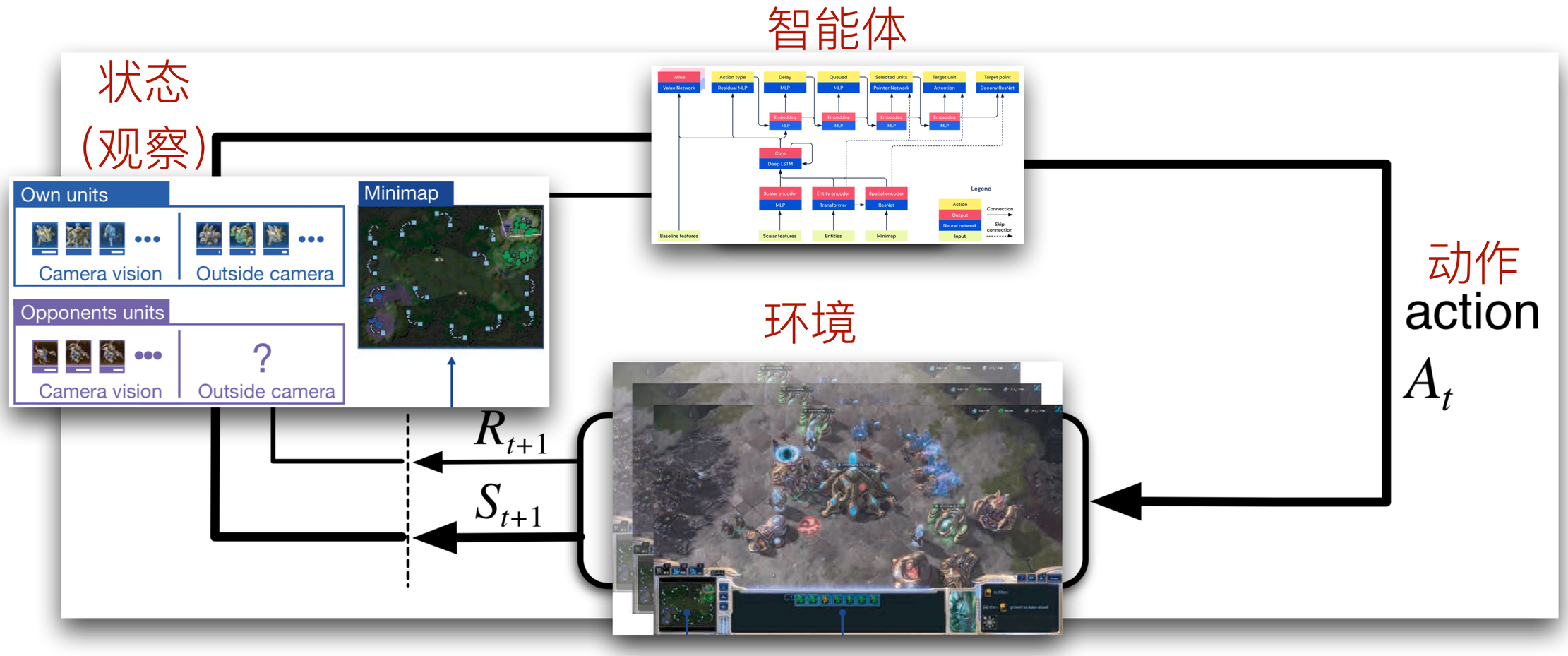
强化学习要素



强化学习要素


状态
(观察)
Own units
Camera vision
Outside camera
Minimap
Opponents units
Camera vision
?
Outside camera
智能体
Agent
环境
动作
action
A_t
R_{t+1}
S_{t+1}

强化学习要素


智能体
状态
(观察)
Own units
Camera vision
Outside camera
Opponents units
Camera vision
?
Outside camera
Minimap
Value
Value Network
Action type
Residual MLP
Delay
MLP
Queued
MLP
Selected units
Pointer Network
Target unit
Attention
Target point
Deconv ResNet
Embedding
MLP
Core
Deep LSTM
Scalar encoder
MLP
Entity encoder
Transformer
Spatial encoder
ResNet
Legend
Action
Output
Neural network
Input
Connection
Skip connection
Baseline features
Scalar features
Entities
Minimap
环境
动作
action
A_t
R_{t+1}
S_{t+1}

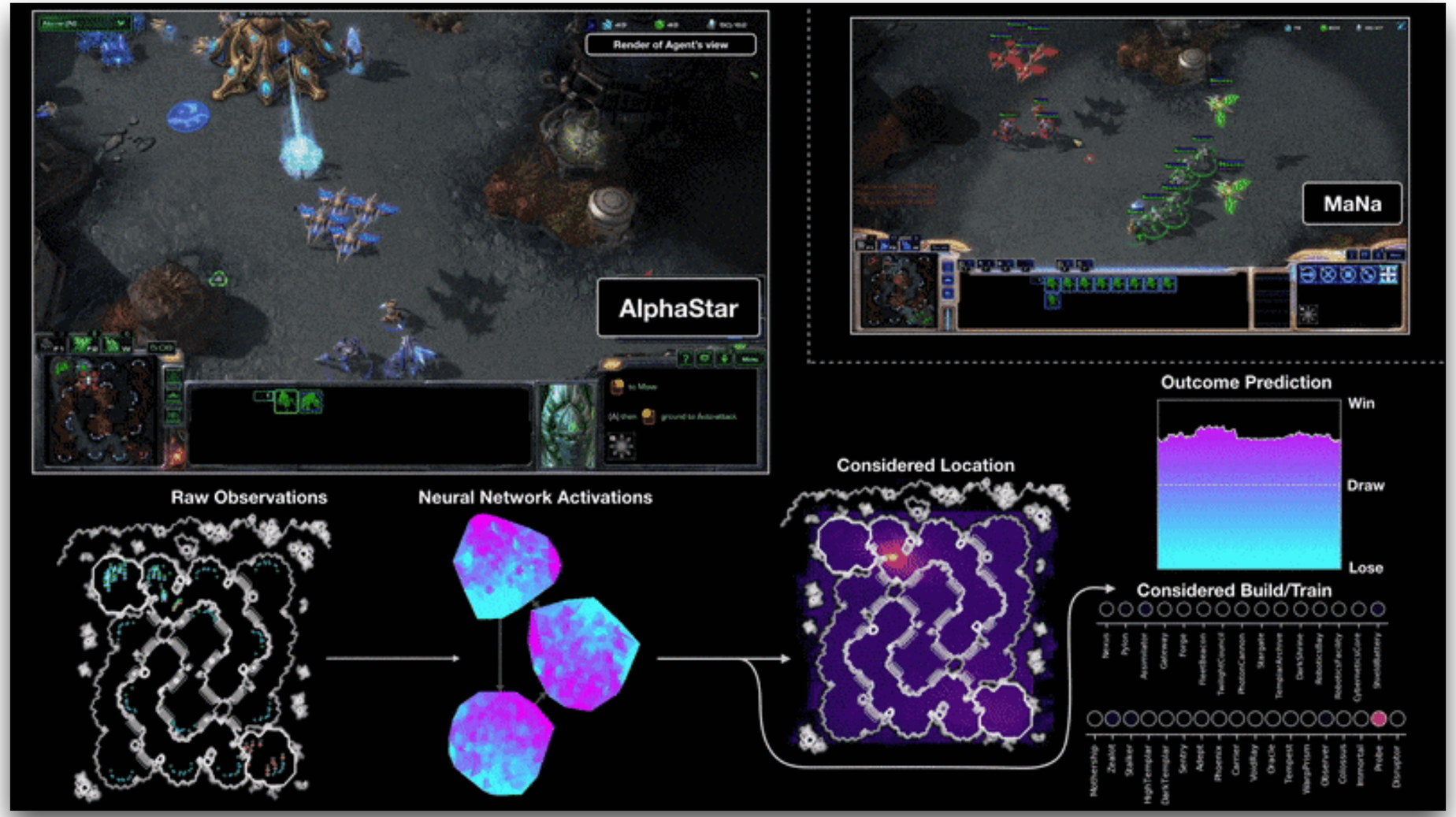
强化学习要素


智能体
状态
(观察)
动作
环境
Own units
Camera vision
Outside camera
Opponents units
Camera vision
?
Outside camera
Minimap
Value
Value Network
Action type
Residual MLP
Delay
MLP
Queued
MLP
Selected units
Pointer Network
Target unit
Attention
Target point
Deconv ResNet
Embedding
MLP
Core
Deep LSTM
Scalar encoder
MLP
Entity encoder
Transformer
Spatial encoder
ResNet
Legend
Action
Output
Neural network
Input
Connection
Skip connection
Baseline features
Scalar features
Entities
Minimap
Action
Move
Attack
Build
What?
Who?
Where?
When next action?
R_{t+1}
S_{t+1}

强化学习要素

环境

状态
(观察)



智能体

动作

AlphaStar环境设计

奖励是什么

奖励	用途	计算	备注
胜负奖励	鼓励胜利	赢+1，平0，输–1	只在最后一刻给
建造顺序(Build order)	模仿人类	编辑距离	人类数据给出参考
建造单位(Built Units)	模仿人类	汉明距离	同上
科技升级(Upgrades)	模仿人类	汉明距离	同上
技能(Effects)	模仿人类	汉明距离	同上

AlphaStar环境设计

奖励是什么

奖励	用途	计算	备注
			刻给
			参考
建造			
科技			
技能(Effects)	模仿人类	汉明距离	同上

编辑距离(Edit Distance)

人类建造顺序是ABCD

智能体的建造顺序是BACD

把BACD变成ABCD所需最小操作是2次

(BACD变成AACD再变成ABCD)

所以建造顺序奖励是-2

AlphaStar环境设计

奖励是什么

奖励	用途	计算	备注
			刻给
			参考
建造			
科技			
技能(Effects)	模仿人类	汉明距离	同上

汉明距离(Hamming Distance)

场上有ABCD四种建筑，各用0或1表示是否建造

人类造了ABC，表示为1110

智能体造了ABD，表示为1101

两者汉明距离为2（两处不同）

“建造单位奖励”为-2

AlphaStar学习算法

强化学习

- 目的：优化策略，使得其期望奖励最大 $J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \sum_{t=0} r(s_t, a_t)$
- 难点：1. 训练的模型不是采样的模型(off-policy), 2. 动作空间高度复杂, 3. 拟合价值函数很难
- 做法：

1. Actor-critic结构，主要基于IMPALA，利用V-trace来训策略网络，并提出UPGO技术帮助训练
2. 利用TD(λ)来训价值网络，并同时输入对手数据
3. 额外引入Supervised Learning Loss和人类统计量Z
4. 多智能体/自学习/League的大规模应用

AlphaStar学习算法

强化学习 – Actor–critic概述

- Actor (演员)：策略网络 $\pi(a_t|s_t)$ 给出当前状态下智能体的动作
- Critic (批评家)：价值网络 $V(s_t) = \mathbb{E}\sum_{t'=t} r_{t'} = \mathbb{E}_{a_t}[r(s_t, a_t) + V(s_{t+1})]$ 判断当前状态下智能体的期望收益
- Actor–critic算法：

1. 在状态s下，计算当前给出动作a相比于“平均动作”获得的优势 (Advantage)

$$A(s_t, a_t) = [r(s_t, a_t) + V(s_{t+1})] - V(s_t)$$

2. 用优势计算策略梯度 (Policy Gradient) $\nabla_\theta J = A(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)$

AlphaStar学习算法

强化学习 – V–trace引入

- Actor–critic算法：
  1. 在状态s下，计算当前给出动作a相比于“平均动作”获得的优势 (Advantage)

$$A^{\pi_\theta}(s_t, a_t) = [r(s_t, a_t) + V^{\pi_\theta}(s_{t+1})] - V^{\pi_\theta}(s_t)$$

  2. 用优势计算策略梯度 (Policy Gradient) $\nabla_\theta J = \mathbb{E}_{\pi_\theta}[A^{\pi_\theta}(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)]$

- 问题：如果正在学习的策略π_θ和负责采集数据 (s_t, a_t) 的策略π_μ不同怎么办?

- 答案：用重要性采样 (Importance Sampling)

$$\nabla_\theta J = \mathbb{E}_{\pi_\mu}[\frac{\pi_\theta(a_t|s_t)}{\pi_\mu(a_t|s_t)} A(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)]$$

AlphaStar学习算法

强化学习 – V–trace

- 策略梯度 (Policy Gradient)

$$\nabla_\theta J = \mathbb{E}_{\pi_\mu}[\frac{\pi_\theta(a_t|s_t)}{\pi_\mu(a_t|s_t)} A^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

- V–trace限制了重要性采样系数

$$\nabla_\theta J = \mathbb{E}_{\pi_\mu}[\rho_t A^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

$$\rho_t = \min(\frac{\pi_\theta(a_t|s_t)}{\pi_\mu(a_t|s_t)}, 1)$$

AlphaStar学习算法

强化学习 – V–trace

- 策略梯度 (Policy Gradient)

$$\nabla_\theta J = \mathbb{E}_{\pi_\mu}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_\mu(a_t|s_t)}\right.$$

IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures

Lasse Espeholt [*1] Hubert Soyer [*1] Remi Munos [*1] Karen Simonyan [1] Volodymyr Mnih [1] Tom Ward [1]
Yotam Doron [1] Vlad Firoiu [1] Tim Harley [1] Iain Dunning [1] Shane Legg [1] Koray Kavukcuoglu [1]

- V–trace限制了重要性采样系数

$$\nabla_\theta J = \mathbb{E}_{\pi_\mu}\left[\rho_t A^{\pi_\theta}(s_t, a_t)\right.$$

$$\rho_t = \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_\mu(a_t|s_t)}, 1\right)$$

4.1. V-trace target

Consider a trajectory $(x_t, a_t, r_t)_{t=s}^{t=s+n}$ generated by the actor following some policy μ. We define the n-steps V-trace target for $V(x_s)$, our value approximation at state x_s, as:

$$v_s \stackrel{\text{def}}{=} V(x_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s}\Big(\prod_{i=s}^{t-1} c_i\Big)\delta_t V, \quad (1)$$

where $\delta_t V \stackrel{\text{def}}{=} \rho_t\big(r_t + \gamma V(x_{t+1}) - V(x_t)\big)$ is a temporal difference for V, and $\rho_t \stackrel{\text{def}}{=} \min\left(\bar{\rho}, \frac{\pi(a_t|x_t)}{\mu(a_t|x_t)}\right)$ and $c_i \stackrel{\text{def}}{=} \min\left(\bar{c}, \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)}\right)$ are truncated importance sampling (IS) weights (we make use of the notation $\prod_{i=s}^{t-1} c_i = 1$ for $s = t$). In addition we assume that the truncation levels are such that $\bar{\rho} \geq \bar{c}$.

AlphaStar学习算法

强化学习 – V–trace

- 策略梯度 (Policy Gradient)

$$\nabla_\theta J = \mathbb{E}_{\pi_\mu}[\frac{\pi_\theta(a_t|s_t)}{\pi_\mu(a_t|s_t)} A^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

- V–trace限制了重要性采样系数

$$\nabla_\theta J = \mathbb{E}_{\pi_\mu}[\rho_t A^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)]$$

$$\rho_t = \min(\frac{\pi_\theta(a_t|s_t)}{\pi_\mu(a_t|s_t)}, 1)$$

AlphaStar学习算法

强化学习 – UPGO 优势估算

- 优势 Advantage

$$A^{\pi_\theta}(s_t, a_t) = [r(s_t, a_t) + V^{\pi_\theta}(s_{t+1})] - V^{\pi_\theta}(s_t)$$

- Upgoing Policy Update (UPGO)

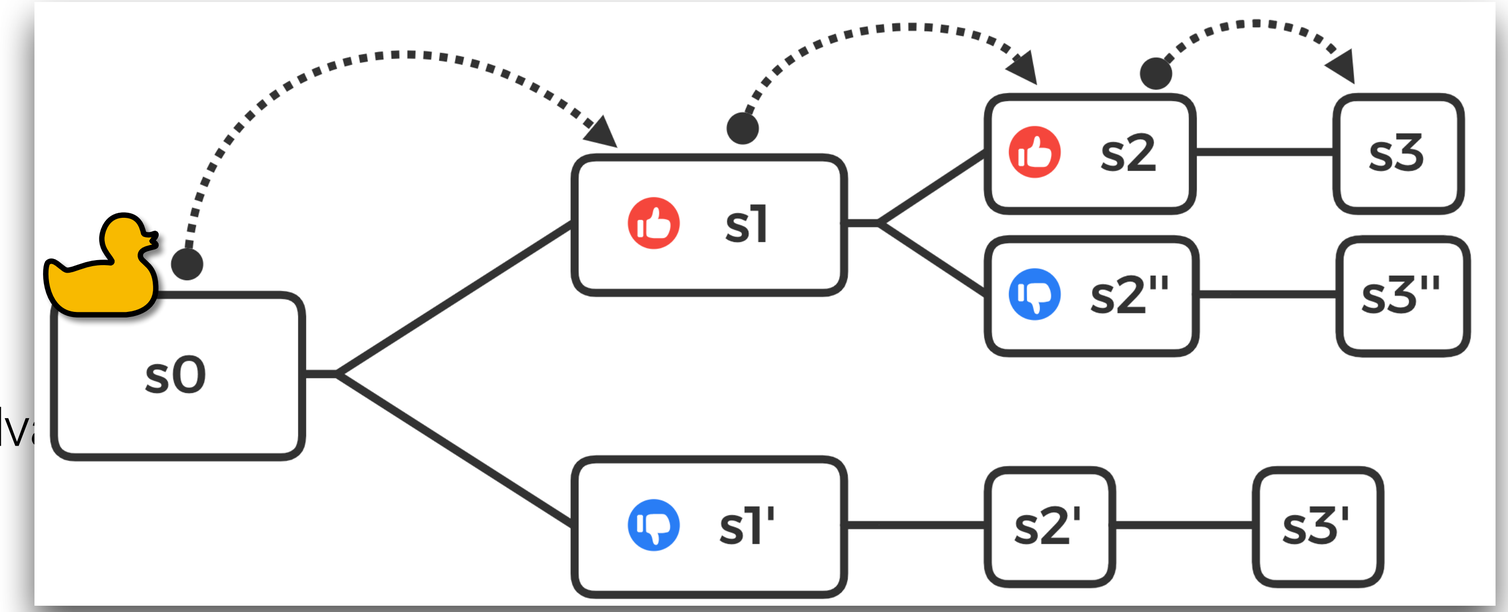
$$A^{\pi_\theta}(s_t, a_t) = G_t^U - V_{\pi_\theta}(s_t)$$

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

$$V(s_t) = \mathbb{E}_{a_t}[r(s_t, a_t) + V(s_{t+1})]$$

$$Q(s_t, a_t) = r(s_t, a_t) + V(s_{t+1})$$

- 优势 Adv



- Upgoing Policy Update (UPGO)

$$A^{\pi_\theta}(s_t, a_t) = G_t^U - V_{\pi_\theta}(s_t)$$

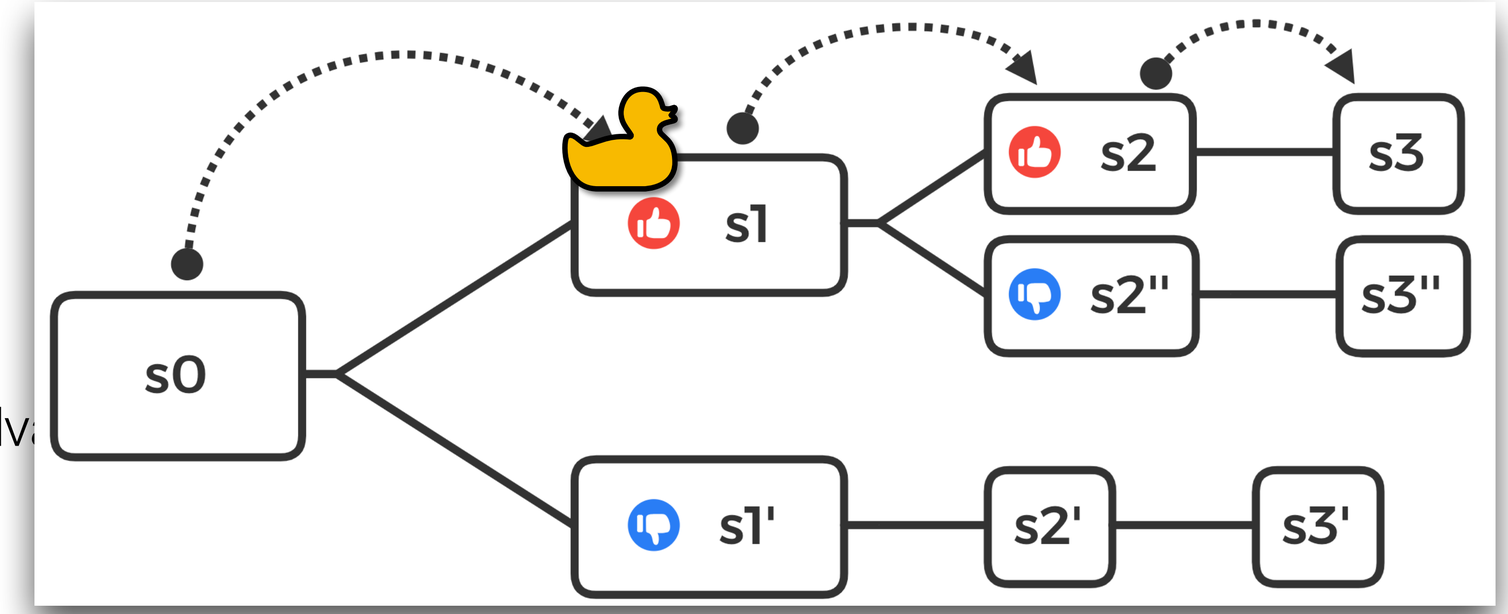
$$G_t^U = \begin{cases} r_t + G_{t+1}^U & \text{if } Q(s_{t+1}, a_{t+1}) \geq V(s_{t+1}) \\ r_t + V(s_{t+1}) & \text{otherwise} \end{cases}$$

若 a_0 是步好棋，此时 G_0 变成 $r_0 + G_1$

$$V(s_t) = \mathbb{E}_{a_t}[r(s_t, a_t) + V(s_{t+1})]$$

$$Q(s_t, a_t) = r(s_t, a_t) + V(s_{t+1})$$

- 优势 Adv



- Upgoing Policy Update (UPGO)

$$A^{\pi_\theta}(s_t, a_t) = G_t^U - V_{\pi_\theta}(s_t)$$

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

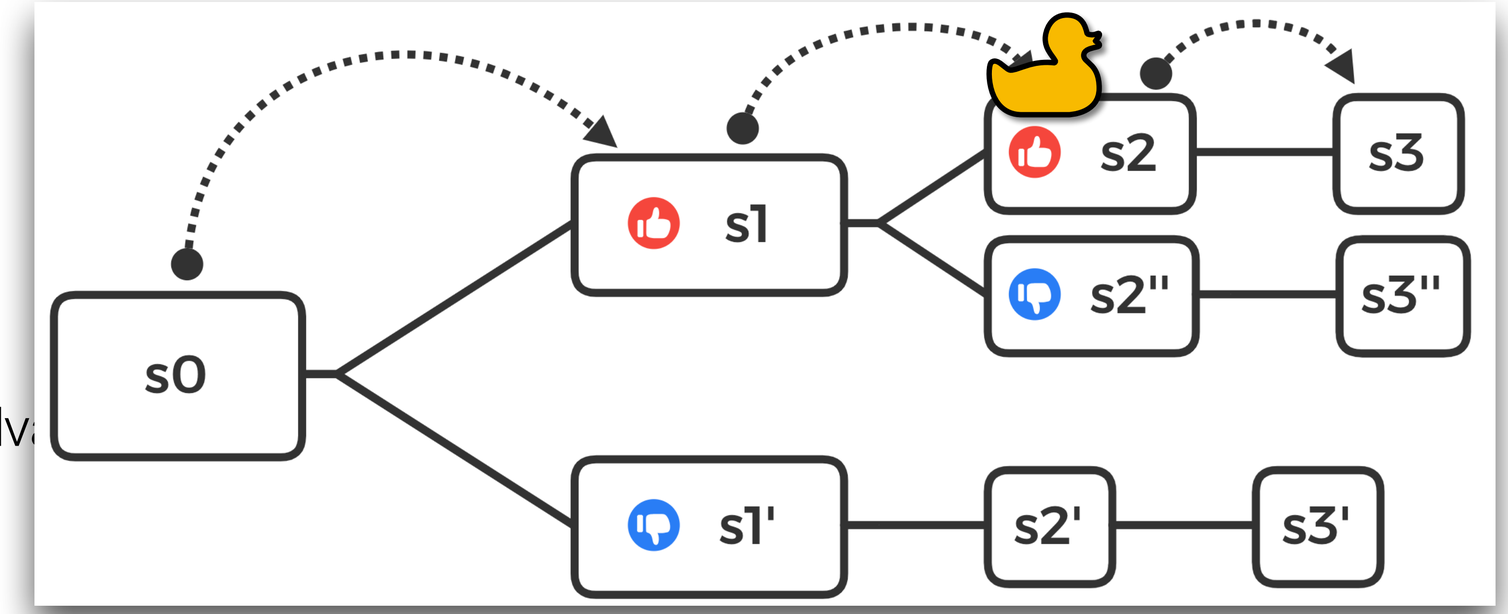
若 a_0 是步好棋，此时 G_0 变成 $r_0 + G_1$

若 a_1 是步好棋，此时 G_0 变成 $r_0 + r_1 + G_2$ ...

$$V(s_t) = \mathbb{E}_{a_t}[r(s_t, a_t) + V(s_{t+1})]$$

$$Q(s_t, a_t) = r(s_t, a_t) + V(s_{t+1})$$

- 优势 Adv



- Upgoing Policy Update (UPGO)

$$A^{\pi_\theta}(s_t, a_t) = G_t^U - V_{\pi_\theta}(s_t)$$

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

若 a_0 是步好棋，此时 G_0 变成 $r_0 + G_1$

若 a_1 是步好棋，此时 G_0 变成 $r_0 + r_1 + G_2$ ...

$$V(s_t) = \mathbb{E}_{a_t}[r(s_t, a_t) + V(s_{t+1})]$$

$$Q(s_t, a_t) = r(s_t, a_t) + V(s_{t+1})$$

个人理解

GAE和UPGO都关心如何将多步以后未来的信息纳入现在的Advantage估计中。GAE使用Soft的形式通过λ项控制未来信息和现在信息的平衡（即偏差和方差的平衡）。UPGO则使用Hard的形式直接将未来乐观的step的信息纳入Advantage。

- Upgoing Policy Update (UPGO)

$$A^{\pi_\theta}(s_t, a_t) = G_t^U - V_{\pi_\theta}(s_t)$$

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

若 a_0 是步好棋，此时 G_0 变成 $r_0 + G_1$

若 a_1 是步好棋，此时 G_0 变成 $r_0 + r_1 + G_2$ ...

$$V(s_t) = \mathbb{E}_{a_t}[r(s_t, a_t) + V(s_{t+1})]$$

$$Q(s_t, a_t) = r(s_t, a_t) + V(s_{t+1})$$

AlphaStar学习算法

强化学习 – TD(λ)

- 价值函数 $V^{\pi_\theta}(s_t) = \mathbb{E}_{\pi_\theta} \sum_{t'=t} \gamma^{t'-t} r(s_t, a_t) = \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)}[r(s_t, a_t) + \gamma V(s_{t+1})]$ 怎么训练?
- TD(0) 法： $L = [(r_t + \gamma V_{t+1}) - V_t]^2$ （圆括号内是向前看0步的结果）
- TD(1)法又称MC法： $L = [(\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}) - V_t]^2$ （圆括号内是向前看∞步的结果）
- TD(λ)：我想用加权平均来平衡向前看0步、1步…、∞步的结果
  1. 已知对于0,1之间的λ有 $(1-\lambda) + (1-\lambda)\lambda + (1-\lambda)\lambda^2 + \ldots = 1$
  2. $R_t = \lim_{T\to\infty}(1-\lambda)(r_t + V_{t+1}) + (1-\lambda)\lambda(r_t + \gamma r_{t+1} + \gamma^2 V_{t+2}) + \ldots + (1-\lambda)\lambda^T(\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} + \gamma^{T-t'+1} V_T)$

参考tfrl的实现：https://github.com/deepmind/trfl/blob/master/trfl/value_ops.py#L159 和Sutton的RL课本12.3节

AlphaStar学习算法

强化学习 – TD(λ)

- TD(λ)：我想用加权平均来平衡向前看0步、1步…、∞步的结果

1. 已知对于0,1之间的λ有 $(1-\lambda)+(1-\lambda)\lambda+(1-\lambda)\lambda^2+\ldots=1$

2. $R_t=\lim_{T\to\infty}(1-\lambda)(r_t+V_{t+1})+(1-\lambda)\lambda(r_t+\gamma r_{t+1}+\gamma^2 V_{t+2})+\ldots+(1-\lambda)\lambda^T(\sum_{t'=t}^{T}\gamma^{t'-t}r_{t'}+\gamma^{T-t'+1}V_T)$

- 可以很简单的迭代计算：

$$R_T=r_T+\gamma V_T$$

For $t=T-1,T-2,\ldots$

$$G_t=r_t+\gamma(1-\lambda)V_t$$

$$R_t=G_t+\gamma\lambda R_{t+1}$$

AlphaStar学习算法

强化学习 – TD(λ)

- TD(λ)：我想用加权平均来平衡向前看0步、1步…、∞步的结果

  1. 已知对于0,1之间的λ有 $(1-\lambda)+(1-\lambda)\lambda+(1-\lambda)\lambda^2+\ldots=1$

  2. $R_t=\lim_{T\to\infty}(1-\lambda)(r_t+V_{t+1})+(1-\lambda)\lambda(r_t+\gamma r_{t+1}+\gamma^2 V_{t+2})+\ldots+(1-\lambda)\lambda^T(\sum_{t'=t}^{T}\gamma^{t'-t}r_{t'}+\gamma^{T-t'+1}V_T)$

- 可以

AlphaStar学习价值网络的两大技巧

a. 价值网络的输入是状态，因此可以不是单个智能体的观察

b. 价值网络仅仅用来计算优势从而优化策略，在实战时不用

c. 价值网络的输出是一个策略在一个状态下的预期收益

1. 将我方智能体(正在学习)及对手的观察一同输入价值网络

2. 学习我方智能体的预期收益，也学习对手的预期收益*

* 注：原文“To reduce variance, during training only, the value function is estimated using information from both the player’s and the opponent’s perspectives. ”

AlphaStar学习算法

强化学习 – 小结

- 目的：优化策略，使得其期望奖励最大 $J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \sum_{t=0} r(s_t, a_t)$
- 难点：1. 训练的模型不是采样的模型(off-policy), 2. 动作空间高度复杂, 3. 拟合价值函数很难
- 做法：

1. Actor-critic结构，主要基于IMPALA，利用V-trace来训策略网络，并提出UPGO技术帮助训练
2. 利用TD(λ)来训价值网络，并同时输入对手数据
3. 额外引入Supervised Learning Loss和人类统计量Z
4. 多智能体/自学习/League的大规模应用

AlphaStar学习算法

模仿学习 – 概述

- 目的：注入人类信息，帮助强化学习
- 做法：
  1. 监督学习预训练得到神经网络初始化
  2. 人类统计量Z用来计算伪奖励函数（pseudo-rewards）
  3. 监督学习loss与强化学习loss一起使用

AlphaStar学习算法

模仿学习 – 人类统计量Z

- 统计量Z实际上是一批数据：

  1. 前20个建造的建筑物和单位

  2. 整局游戏中建造的单位、建筑，升级的科技和技能（所以可以用布尔向量表示）

- 统计量Z作为神经网络的输入传给策略网络和价值网络（称为conditioned on Z）

- 除了胜负奖励以外的其他奖励是即时发送的还是终局才发送的文章没有明说

奖励	用途	计算
胜负奖励	鼓励胜利	赢+1，平0，输–1
建造顺序(Build order)	模仿人类	编辑距离
建造单位(Built Units)	模仿人类	汉明距离
科技升级(Upgrades)	模仿人类	汉明距离
技能(Effects)	模仿人类	汉明距离

AlphaStar学习算法

模仿学习 – 人类统计量Z

- 统计量Z实际上是一批数据：
  1. 前
  2. 整
- 统计量
  （称为
- 除了胜负奖励以外的其他奖励是即时发送的还是终局才发送的文章没有明说

编辑距离(Edit Distance)

人类建造顺序是ABCD

智能体的建造顺序是BACD

把BACD变成ABCD所需最小操作是2次

(BACD变成AACD再变成ABCD)

所以建造顺序奖励是-2

计算
平0，输-1
辑距离
明距离
明距离
明距离

AlphaStar学习算法

模仿学习 – 人类统计量Z

- 统计量Z实际上是一批数据：
  1. 前
  2. 整
- 统计量
  （称为
- 除了胜负奖励以外的其他奖励是即时发送的还是终局才发送的文章没有明说

汉明距离(Hamming Distance)

场上有ABCD四种建筑，各用0或1表示是否建造

人类造了ABC，表示为1110

智能体造了ABD，表示为1101

两者汉明距离为2（两处不同）

“建造单位奖励”为–2

计算
平0，输–1
辑距离
明距离
明距离
明距离

AlphaStar学习算法

模仿学习

- 在监督学习中，有10%的概率置零Z
- 每个伪奖励函数有75%的概率置零，各自有一个独立的价值网络和loss函数

- 人类信息的利用：

1. 有监督学习预训练得到神经网络初始化
2. 人类统计量Z用来计算伪奖励函数（pseudo-rewards）
3. 有监督学习loss与强化学习loss一起使用

AlphaStar学习算法

自学习 (Self-play)

- 一个被忽视的问题：对手是谁?
- 自学习 (Self-play)：自己和自己打，AlphaGo的重要技术，AlphaStar做了很多拓展
- Self-play的问题：策略循环
- 虚拟自学习Fictitious Self-Play（FSP）：训练过程中，每隔一段时间给自己存档，得到一个种群。均匀随机地从种群中选出对手与正在训练的智能体对战。
- FSP的问题：智能体与太菜的对手打浪费时间（胜率总是100%）

AlphaStar学习算法

自学习 (Self-play) – Prioritized Fictitious Self-play (PFSP)

- FSP的问题：智能体与太菜的对手打浪费时间（胜率总是100%）
- AlphaStar提出有优先级的虚拟自学习 (Prioritized Fictitious Self-play)

1. 按照胜率确定从种群中挑对手的概率，常打败智能体的对手（高手）概率高，反之亦然。

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

2. 式子中f为加权函数，AlphaStar对不同“类型”的智能体选取了不同的f

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

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

AlphaStar学习算法

联盟训练 League Training

- 在Self-play (SP)中，你的对手只有一个，就是最新的自己
- 在FSP/PFSP中，你的对手是一个种群，其中每个智能体都是你的祖先
- AlphaStar称你的“对手池”为联盟 (league)，并将其智能体分为以下三类：

1. 主智能体 (Main Agent)：正在训练的智能体及其祖先，重点培养对象
2. 联盟利用者 (League Exploiter)：能打败联盟里的所有智能体，它发现了全局盲点
3. 主利用者 (Main Exploiter)：能打败正在训练的主智能体，它发现了他们的弱点

Exploiter n. 剥削者/利用者

实际上“explore and exploit”是强化学习的重要概念。exploit在这里指的是“利用”的意思。具体指的是智能体一直采用期望回报最大的动作。与之相对的，explore指的是“探索”，即智能体可以采用并非最大化期望回报的动作以尝试新的动作以探索环境。

- *exploit, v., to use sth well in order to gain as much from it as possible — Oxford Advanced Learners Dictionary 7th edition*

- AlphaStar称你的“对手池”为联盟 (league)，并将其智能体分为以下三类：

1. 主智能体 (Main Agent)：正在训练的智能体及其祖先，重点培养对象
2. 联盟利用者 (League Exploiter)：能打败联盟里的所有智能体，它发现了全局盲点
3. 主利用者 (Main Exploiter)：能打败正在训练的主智能体，它发现了他们的弱点

AlphaStar学习算法

联盟训练 League Training

- AlphaStar称你的“对手池”为联盟 (league)，并将其智能体分为以下三类：
  1. 主智能体 (Main Agent)，联盟利用者 (League Exploiter)，主利用者 (Main Exploiter)
  2. 他们的区别只有一下三个部分
     1. 如何选取训练过程中对战的对象
     2. 在什么情况下存档 (snapshot) 现在的策略
     3. 以多大的概率将策略的参数重设为监督训练给出的初始化

AlphaStar学习算法

联盟训练 – 主智能体

- 主智能体指正在训练的智能体及其祖先
- 场上只有3个正在学习的主智能体 (每个种族一个)
- 主智能体的祖先也会加入league中然后进行有关的自学习

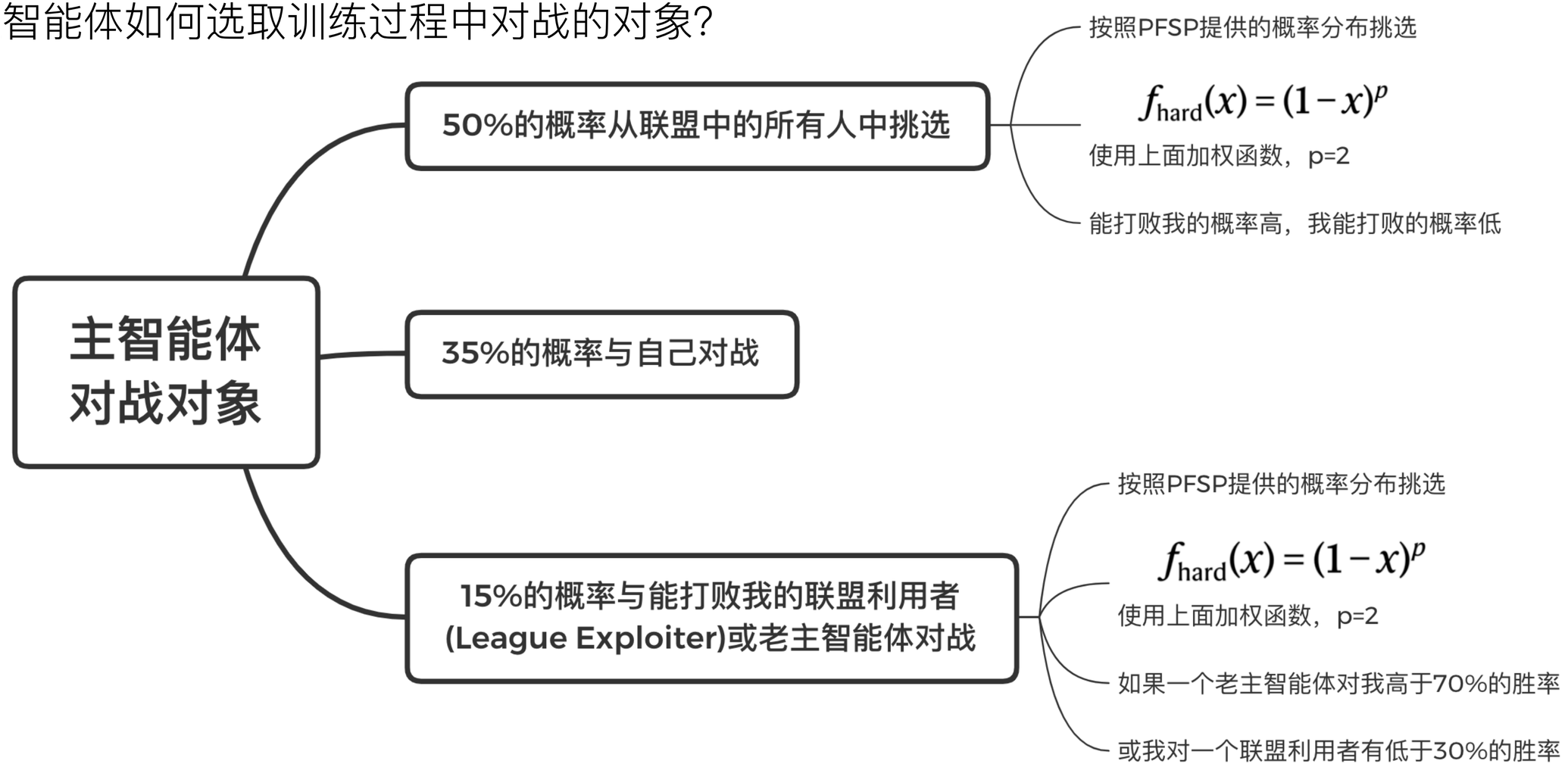
1. 如何选取训练过程中对战的对象
2. 在什么情况下存档 (snapshot) 现在的策略
3. 以多大的概率将策略的参数重设为监督训练给出的初始化

AlphaStar学习算法

联盟训练 – 主智能体

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

- 主智能体如何选取训练过程中对战的对象?



AlphaStar学习算法

联盟训练 – 主智能体 (Main Agent)

- 主智能体指正在训练的智能体及其祖先
- 场上只有3个正在学习的主智能体 (每个种族一个)
- 主智能体的祖先也会加入league中然后进行有关的自学习

1. 如何选取训练过程中对战的对象
2. 在什么情况下存档 (snapshot) 现在的策略：每隔2×10^9个时间步之后就存
3. 以多大的概率将策略的参数重设为监督训练给出的初始化：永不重设

AlphaStar学习算法

联盟训练 – 联盟利用者 (League Exploiter)

- 联盟利用者指能打败联盟里的所有智能体的智能体及其祖先
- 场上有6个正在学习的联盟利用者 (每个种族2个)
- 主智能体的祖先也会加入league中然后进行有关的自学习

1. 如何选取训练过程中对战的对象：按照PFSP给的概率与全联盟的对手对战
2. 在什么情况下存档 (snapshot) 现在的策略：以70%胜率打败联盟中的所有人，或者距上次存档2×10^9个时间步之后就存
3. 以多大的概率将策略的参数重设为监督训练给出的初始化：在存档的时候，有25%概率把场上的联盟利用者的策略重设成监督学习给出的初始化

AlphaStar学习算法

联盟训练 – 主利用者 (Main Exploiter)

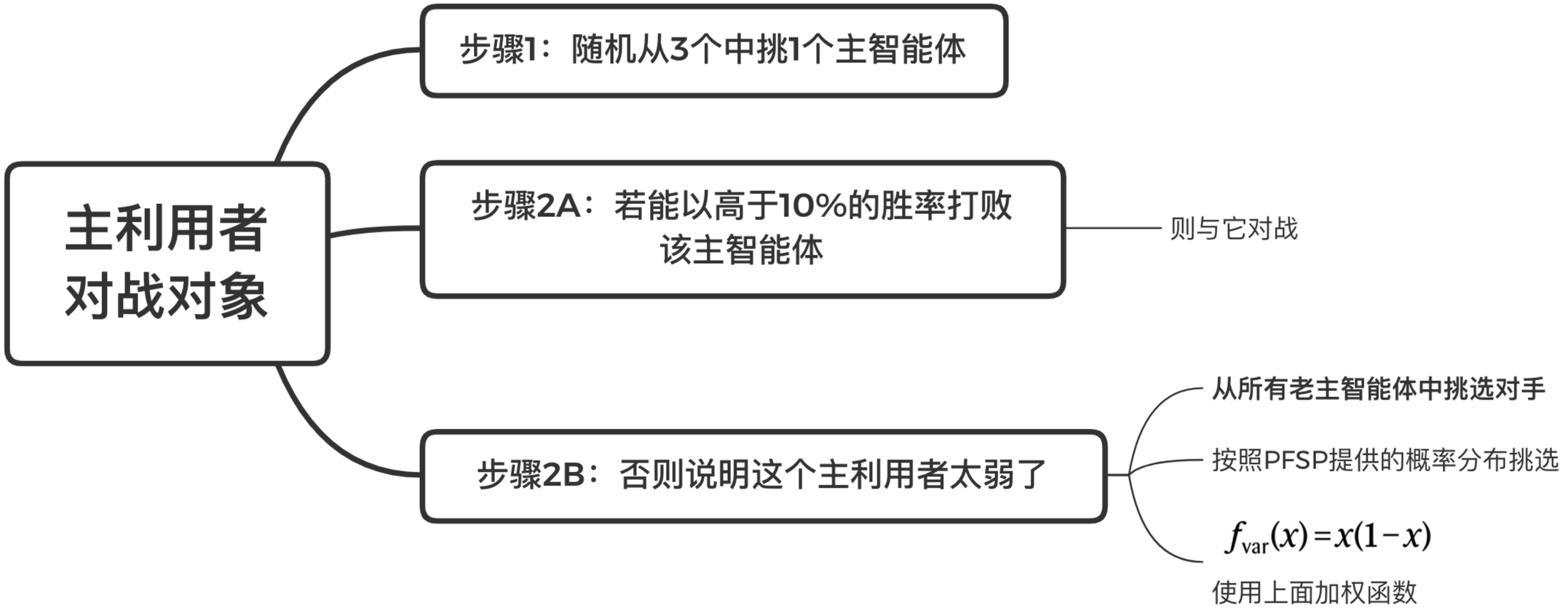
- 主利用者指能打败正在训练的主智能体的智能体及其祖先
- 场上有3个正在学习的主利用者 (每个种族一个)
- 主智能体的祖先也会加入league中然后进行有关的自学习

1. 如何选取训练过程中对战的对象
2. 在什么情况下存档 (snapshot) 现在的策略
3. 以多大的概率将策略的参数重设为监督训练给出的初始化

AlphaStar学习算法

联盟训练 – 主利用者 (Main Exploiter)

- 如何选取训练过程中对战的对象



AlphaStar学习算法

联盟训练 – 主利用者 (Main Exploiter)

- 主利用者指能打败正在训练的主智能体的智能体及其祖先
- 场上有一个正在学习的主利用者(每个种族一个)
- 主智能体的祖先也会加入league中然后进行有关的自学习

1. 如何选取训练过程中对战的对象
2. 在什么情况下存档 (snapshot) 现在的策略：以70%的胜率打败全部三个正在学习的策略主智能体，或者距上次存档4×10^9个时间步之后就存
3. 以多大的概率将策略的参数重设为监督训练给出的初始化：每次存档之后就重设初始化

AlphaStar学习算法

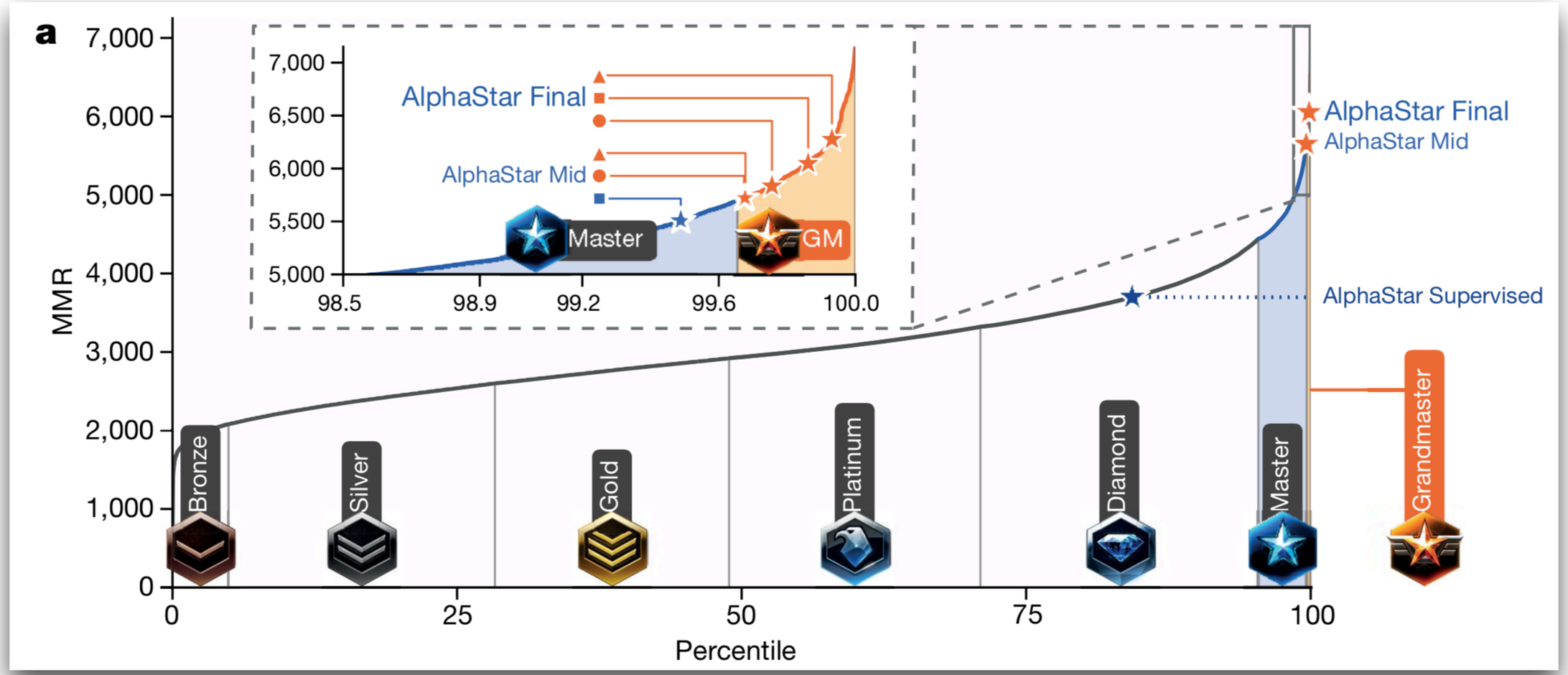
联盟训练 League Training

- 在Self-play (SP)中，你的对手只有一个，就是最新的自己
- 在FSP/PFSP中，你的对手是一个种群，其中每个智能体都是你的祖先
- AlphaStar称你的“对手池”为联盟 (league)，并将其智能体分为以下三类：

1. 主智能体 (Main Agent)：正在训练的智能体及其祖先，重点培养对象
2. 联盟利用者 (League Exploiter)：能打败联盟里的所有智能体，它发现了全局盲点
3. 主利用者 (Main Exploiter)：能打败正在训练的智能体，它发现了他们的弱点

训练过程与结果

主要结果



- x: 天梯排名，y：MMR (match making rating) 分数

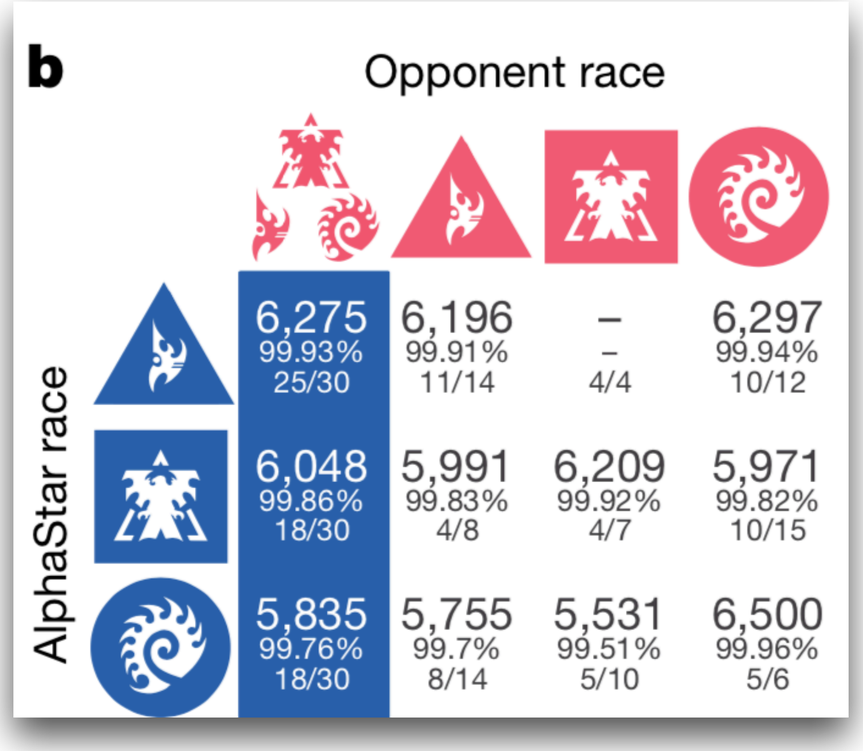
训练过程与结果

主要结果

星灵

人族

异虫



- 从左至右：星灵Protoss，人族Terran，异虫Zerg

训练过程与结果

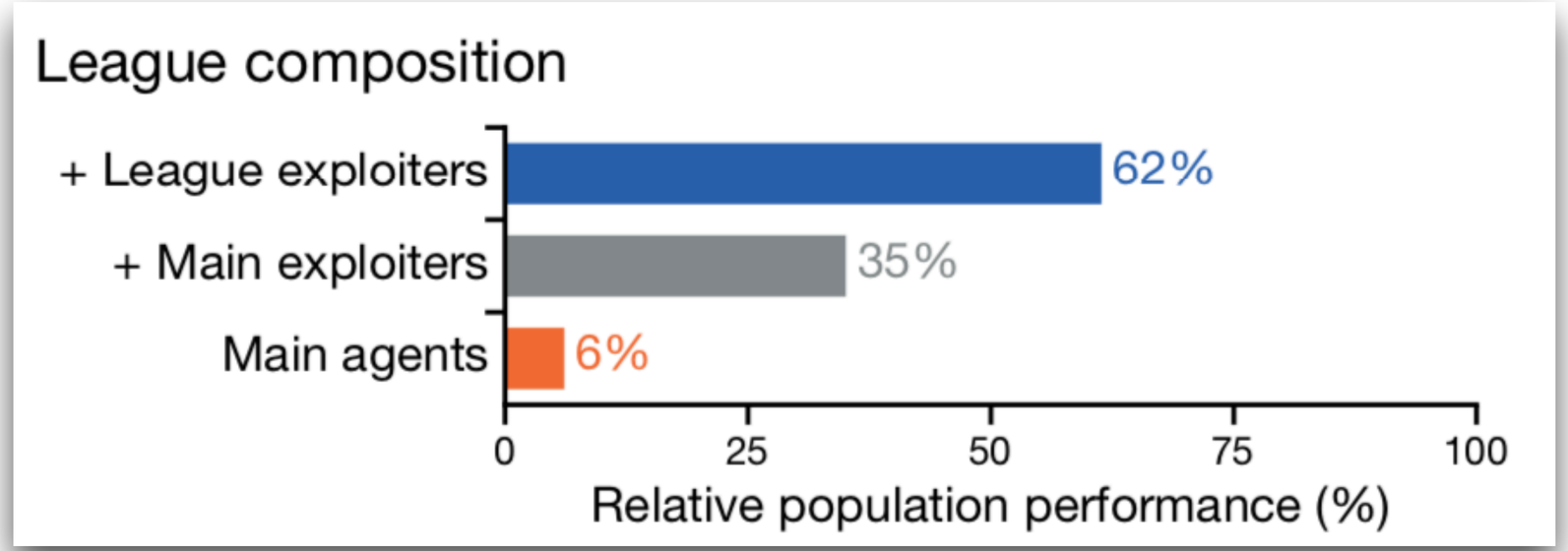
消融实验 – Ablation Study

联盟的构成

再加联盟利用者

添加主利用者

只有主智能体



训练过程与结果

消融实验 – Ablation Study

人类数据使用

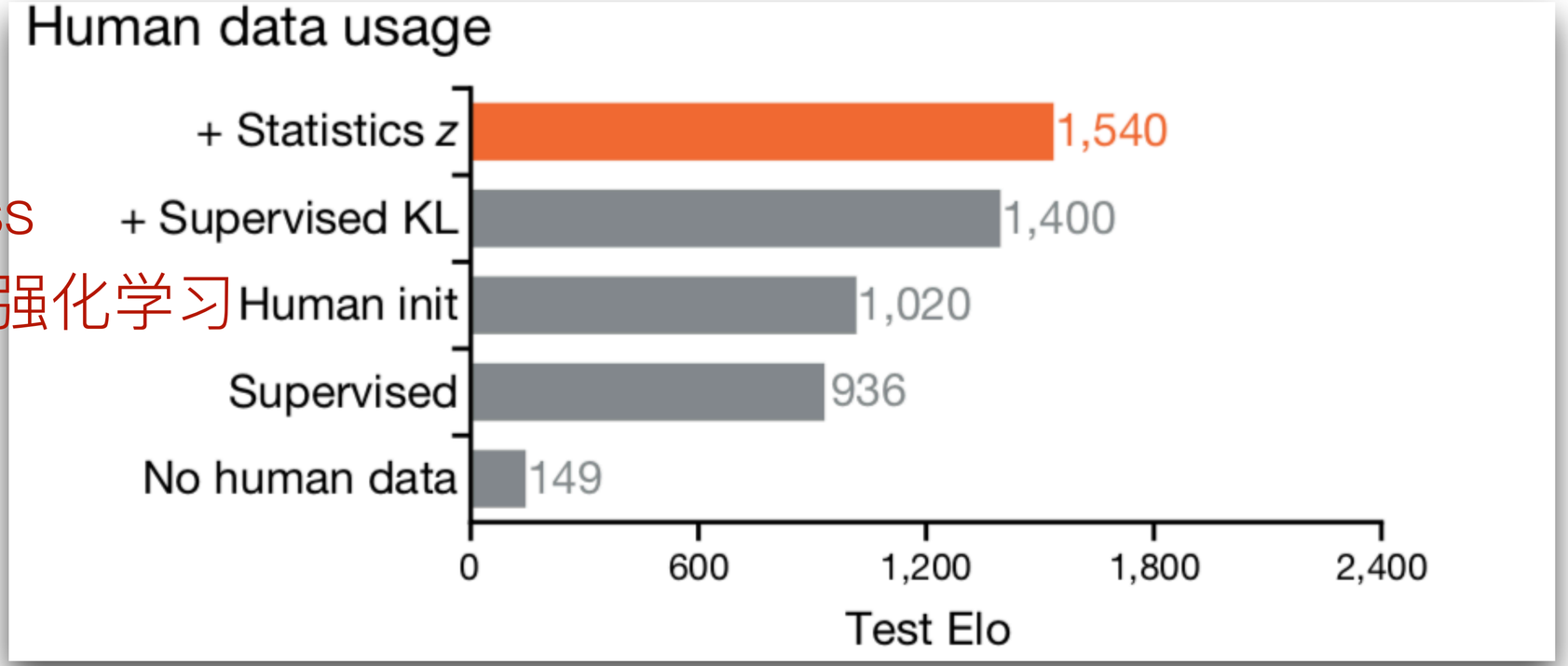
引入人类统计量Z

强化学习再加监督loss

使用人类初始化后的强化学习

纯监督学习

无人类数据



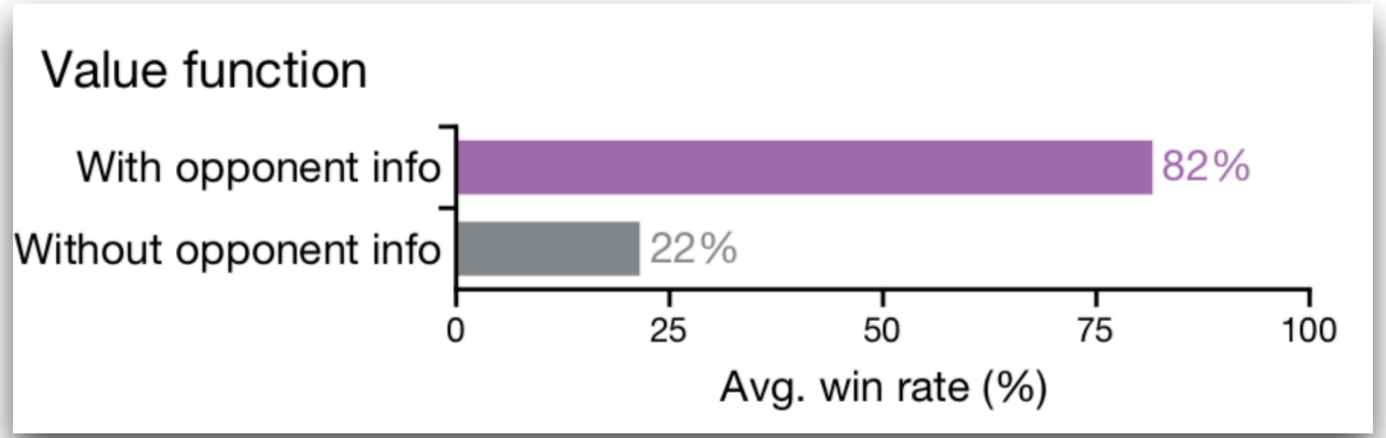
训练过程与结果

消融实验 – Ablation Study

价值函数

引入对手观察

无对手观察



总结

1. 高度复杂的神经网络融合了列表、图像、标量信息等输入
2. 通过Autoregressive的网络设计解耦了结构化的动作空间
3. 模仿学习和监督学习的成功运用（统计量Z等）
4. 复杂的强化学习算法
5. 复杂的联盟训练策略
6. 大量的计算资源

谢谢！

致谢

周博磊教授

周航

牛雅哲

郭悦

参考文献

1. Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.
2. Espeholt, Lasse, et al. "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures." arXiv preprint arXiv:1802.01561 (2018).
3. Vinyals, Oriol, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." Nature 575.7782 (2019): 350-354.
4. Schulman, John, et al. "High-dimensional continuous control using generalized advantage estimation." arXiv preprint arXiv:1506.02438 (2015).
5. tfrl代码库：https://github.com/deepmind/trfl
6. Heinrich, Johannes, Marc Lanctot, and David Silver. "Fictitious self-play in extensive-form games." International Conference on Machine Learning. 2015.
7. Oxford Advanced Learners Dictionary 7th edition