通过计算机视觉及离线强化学习游玩非嵌入式卡牌类即时战略游戏

0.1 摘要

人工智能在游戏中的研究已经取得了很大的进展，例如棋盘游戏、MOBA 游戏、RTS 游戏。然而目前复杂的智能体均为嵌入式开发，即智能体可以直接从游戏底层获取状态信息，与人类玩家所获取的图像信息完全不同，图像信息中往往带有很多噪声，而这些可能干扰玩家决策，这导致了竞技的不公平性产生，因此复杂的非嵌入式智能体开发仍然是一个挑战。此外，卡牌类 RTS 游戏具有复杂的卡牌特征与巨大的状态空间等难点，仍然缺乏解决方案。我们提出了一种基于非嵌入式离线强化学习的训练策略，通过图像信息输入在即时策略游戏皇室战争（Clash Royale）[①]中实现实时的自主博弈对局。由于当前没有关于该游戏的目标识别数据集，因此我们设计了一种生成式数据集用于训练目标识别模型，我们结合当前目标识别与光学文本识别的前沿算法对图像信息进行特征提取，并使用离线强化学习算法进行训练，实现了移动设备上的实时图像获取、感知特征融合、智能体决策及设备控制，能够与对手实时对局，并成功战胜了游戏中的内置 AI。本文的全部代码均已开源 `https://github.com/wty-yy/katacr`。

关键词：目标识别；光学文本识别；离线强化学习

0.2 简介

即时战略游戏（Real-Time Strategy，RTS）通常具有巨大的状态空间、稀疏奖励、信息不完全和策略多样性等特点，卡牌类 RTS 游戏是一种特殊的 RTS 游戏，两名玩家使用手牌进行实时决策对局。与一般的卡牌类游戏不同，卡牌类 RTS 游戏与复杂的卡牌特性有关，也和卡牌的选择与使用的时机有关，卡牌可以在战场中的不同位置使用，这产生了巨大的状态空间。玩家无法看到其他玩家的手牌，这产生了不完整的信息。游戏结果需要到对局结束时才能知道是否胜利，因此游戏中的奖励十分稀疏。进攻、防守、灵活拉扯、控制节奏等策略使比赛具有很大的不确定性。

自从 Mnih 等人 [1] 在 Atari 游戏上成功使用深度强化学习（Deep reinforcement learning, DRL）算法超越人类在该游戏上平均水平后，研究人员们尝试将 DRL 算法运用于各种游戏中，例如：棋盘游戏（围棋中 Silver 等人 [2] 的 AlphaGO 和 Vinyals 等人 [3] 的 AlphaStar），MOBA 游戏（DOTA2 中 Berner 等人 [4] 的 OpenAI Five），RTS 游戏（星际争霸中 Vinyals 等人 [3] 的 AlphaStar）。值得注意的是，在复杂游戏，例如 DOTA2 和星际争霸中的智能体均为嵌入式模型，即智能体可以直接从游戏底层直接获取到准确的状态信息，而这种获取游戏状态的方式与人类完全不同。人类主要通过图像信息理解游戏状态，但图像信息中通常伴有噪声，这些噪声可能干扰玩家决策，因此不同的状态获取方式会导致游戏竞技上的不公平性产生。另一方面，嵌入式智能体能够高效地与环境进行交互，所以可以使用在线强化学习算法训练智能体，例如 Mnih 等人 [1] 的 Deep Q Network（DQN）算法，Schulman 等人 [5] 的 Proximal Policy Optimization（PPO）算法。非嵌入式智能体通常与环境交互速度十分缓慢，这也导致在线训练方法成为一大挑战，而离线强化学习（Offline reinforcement learning）是解决这一类问题的一种方法，例如 Kumar 等人 [6] 的 Conservative Q-Learning（CQL），Lili Chen 等人 [7] 的 Decision Transformer（DT）。这些算法通过学习专家数据集，能够在不和环境进行交互情况下学习策略，并最终与真实环境进行交互验证模型性能。

[①]Clash Roayle 是 Supcell 在芬兰或其他国家的商标，本文中仍和内容均不应视为 Supercell 的批准或赞助。

本文我们只关注一款流行的卡牌类 RTS 游戏，皇室战争（Clash Royale, CR[②]）。具体来说，我们设计了如图0-1所示的一整套非嵌入式智能体游玩 CR 的交互流程，包括与移动设备的交互，对图像特征的感知融合，以及智能体决策模型的控制。本文通过收集专家数据集，在这种奖励稀疏，且智能体和环境交互速度缓慢的情况下，使用离线强化学习算法训练智能体，虽然智能体在训练中没有与真实环境进行过交互，但是最终却能成功战胜真实游戏中内置的顶级 AI。

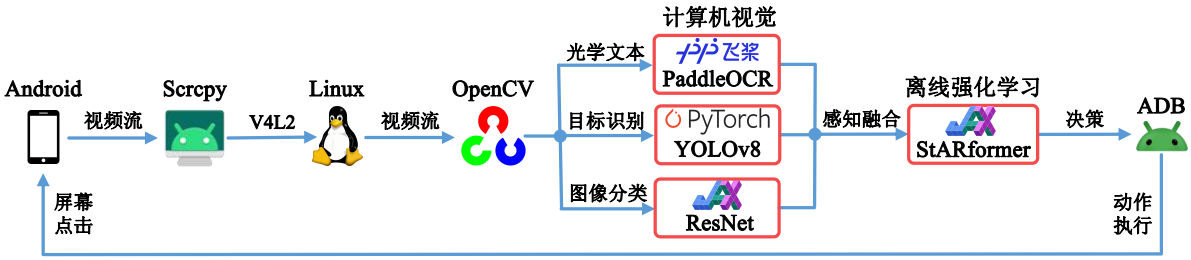


图 0-1 Information flow transmission flow chart

0.3 卡牌即时策略游戏

皇室战争是一款流行的卡牌类 RTS 游戏，本文仅考虑经典的一对一对战模式，双方玩家需要通过实时决策部署手牌来战胜对手，我们测试的游戏版本为 2024 年 5 月第 59 赛季。

0.3.1 感知场景

在对局过程中，游戏场景如图 0-2 所示，本文所需的感知内容主要分为如下四个主要部分（竞技场，手牌，游戏时间，总圣水）和竞技场中的三个子部分（主塔，副塔，部队）。

竞技场（Arena）中包括防御塔及双方所部署的部队和建筑物。竞技场的右上方是当前游戏时间，表示当前阶段所剩的时间。竞技场的下方是当前手牌和总圣水，手牌包括当前卡牌图像及其所需的圣水费用，卡牌包含部队、法术、建筑三种，部队分为空中与地面两种。部队是在竞技场中可移动单位，能够攻击与防御；法术是一种区域效果，通常可以迅速杀死一群部队；建筑物使用后就被固定在一个位置，攻击在其附近的部队。总圣水是使用卡牌所需花费的材料，并会随着时间自动回复。

玩家需要在游戏时间内，尽可能多的推掉敌方的防御塔，在下面小结中，我们将具体介绍游戏的过程，并对游戏状态进行数学建模。

0.3.2 游戏过程

游戏过程包括卡牌轮换和获胜条件两个部分，卡牌轮换是指玩家的手牌在使用后的变换规律，获胜条件是指在对局结束时判定玩家胜利、平局、失败的条件。

0.3.2.0.1 **卡牌轮换** 每次对局中双方玩家牌库大小均为 8 张，在对局开始时，随机在队列中初始化卡牌的出现顺序，每次从队首取出 4 张手牌，当玩家使用卡牌后，使用完的卡牌将被重新加入队尾，所以理论上当一方使用完 8 张不同卡牌时，可以通过逻辑计算出未出现卡牌类

[②] https://clashroyale.com/


Time left:
0:14
剩余时间
x2
敌方地面部队
敌方主塔
敌方副塔
敌方空中部队
我方地面部队
我方地面部队
我方地面建筑
我方副塔
我方主塔
可用手牌
Next:
Max:10
可用圣水

图 0.2.1

别。玩家需要基于当前竞技场上的实时状态、手牌类别及总圣水信息，实时决策卡牌类别的使用位置，采取进攻或防守策略。

0.3.2.0.2 **获胜条件** 每位玩家的目标是摧毁尽可能多的敌方防御塔，优先摧毁敌方主塔的玩家立刻获胜。游戏中在两种阶段下存在不同的胜利条件：

1. 常规时间：持续三分钟，目标为摧毁更多的敌方防御塔，若时间结束时防御塔数量一致，则进入加时赛，否则防御塔多的一方获胜。
2. 加时赛：持续两分钟，在该阶段中，首个摧毁敌方剩余防御塔的玩家获胜，若加时赛结束时仍未分出胜负，则比较双方防御塔的最低生命值，最低生命值较高者获胜，若仍未分出胜负，则为平局。

0.3.3 游戏状态

对于竞技场（Arena）而言，其防御塔、部队的状态由位置、生命值、类别和派别四种参数构成。

1. **位置** 对于第 i 个单位的坐标，将其记为二维离散坐标 $\boldsymbol{x}_i$，表示单位所处于大小 18×32 的网格内。
2. **类别** 对于第 i 个单位的类别，将其记为一维离散值 c_i，这是每个部队或建筑的唯一标签，可选类别范围从 1 到 150。
3. **派别** 对于第 i 个单位的派别，将其记为二值化离散值 bel_i，表示单位所从属方（ownship），取值仅在 0 和 1 之中，分别代表我方和敌方单位。
4. **生命值** 对于第 i 个单位的生命值，将其记为黑白条状图像 bar_i，在条状图像中记录了每个单位所剩余的生命值。

对于游戏时间，可以通过当前剩余时间换算出经过的总时间 t，一场对局的总时间不超过 300 秒，因此时间的取值范围为 $0 \leqslant t \leqslant 300$。对于手牌信息，当每次使用手牌后，手牌位为会出现 1 秒的空置时间，当手牌位非空时，手牌信息由每个位置的手牌类别构成，记为四维离散坐标 card，由于每个玩家最多携带 8 张手牌，因此每个维度上数值范围在 1 到 8 之间。总圣水信息记为一维离散值 elixir，由于总圣水上限为 10，因此圣水的取值范围为 $0 \leqslant \text{elixir} \leqslant 10$。

0.4 生成式数据集

对于图0-2竞技场中第 i 个单位 u_i，由 0.3.3 中定义，u_i 可以表示为 $(\boldsymbol{x}_i, c_i, bel_i, bar_i)$，本章节关注于如何建立一个生成式数据集包含 $\boldsymbol{x}_i, c_i, bel_i$ 信息。

由于训练目标识别模型需要大量带边界框标签的图像，而目前没有任何有关该游戏的目标识别数据集，如果人工逐帧标记低效且成本高昂，因此我们提出一种高效的带标签图像生成方案，并在重构后的 YOLOv8 模型上训练，再在真实视频流数据集验证集中进行验证，得到了如表??所示的极高准确率，验证了生成式数据集的可行性。

生成式数据集需要基于每个类别单位的切片图像，切片图像与识别模型的更新流程如图0-3所示，其中左上部分的“原视频流”为一个回合的视频数据，若存在之前训练的目标识别模型，则使用该模型进行对视频流进行辅助标记，从而获得“辅助标记视频流”，再进行手工标记，否则直接对“原视频流”进行手工标记；在手工标记中，以 0.5 秒作为标记间隔，进

行人工目标框标记；然后将目标框和原图像同时传入到 SAM 模型当中获取前景分割，将分割后的结果进行人工筛选获得“切片数据集”；基于已完成的切片数据集，利用生成式数据集算法，对目标识别模型进行迭代更新，从而用于下一次的辅助标记过程。

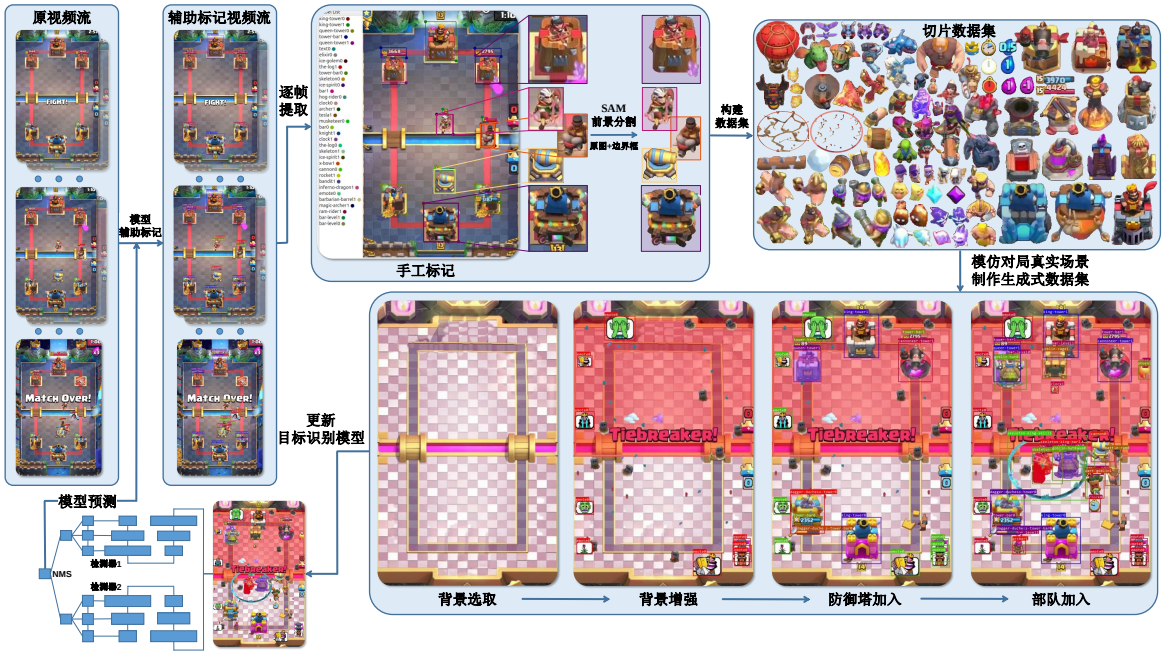


图 0-3 目标识别生成式数据集制作流程

0.4.1 生成算法

假设将每个切片作为绘制单位，第 i 个生成对象定义为 $g_i = (img_i, box_i, level_i)$，其中 img_i 为第 i 个对象对应的切片图像；box_i 为第 i 个对象对应的边界框，可以表示为 (x, y, w, h, c, bel)，其中 (x, y) 为切片中心点在整个图片中的坐标，(w, h) 为切片宽高，c 为当前切片类别，bel 为当前切片的所属派别；$level$ 为当前切片的所属图层等级，生成切片时按照图层从低到高依次生成切片，图层的划分如表 0-1 所示。

表 0-1 图层等级与切片类别关系表

图层等级	切片类别
0	地面法术，地面背景部件
1	地面部队，防御塔
2	空中部队，空中法术
3	其余待识别部件，空中背景部件

绘制单位的插入流程如图 0-3 中右下角部分所示，具体细节如下：

1. 背景选取：从数据集中随机选取一个去除防御塔、部队及文本信息的空竞技场作为背景图片。
2. 背景增强：加入背景板中的非目标识别部件，用于数据增强，例如：部队阵亡时的圣水，场景中随机出现的蝴蝶、花朵等。

3. 防御塔加入：在双方的三个防御塔固定点位上随机生成完好或被摧毁的防御塔，并随机选取生成与之相关联的生命值信息。
4. 部队加入：按照类别出现次数的反比例 $\left\{\frac{1}{n_{c_i}-n_{\min}+1}\right\}_{i=1}^{|C|}$ 所对应的分布进行类别随机选取，其中 $n_{c_i},(c_i \in C)$ 表示类别 c_i 的切片之前生成的总次数，$n_{\min}=\min\{n_{c_i}\}_{i=1}^{|C|}$；在竞技场中按照动态概率分布（具体见附录 ??）随机选择生成点位，并随机选取生成与之相关的等级、生命值、圣水、时钟等信息。

输入: 绘制单位序列 $U=\{u_i\}$，覆盖率阈值 α，待识别类别集合 C
输出: image, box

image ← 空图像，box ← {} // 初始化参数
$U \leftarrow \{u_i \in U : u_i^{level} > u_j^{level}, \forall i,j \in \{1,\cdots,|U|\}$ 且 $i<j\}$
while True do
  mask ← 空掩码，$U_{avail} \leftarrow U$
  for $i=1,2,\cdots,|U|$ do
    if $\frac{u_i^{img} \cap \text{mask}}{u_i^{img}} > \alpha$ then
      $U_{avail} \leftarrow U_{avail} - R(u_i)$ // 删除与 u_i 相关联的单位
    end
    mask ← mask $\cup\, u_i^{img}$
  end
  if $|U_{avail}| = |U|$ then
    break // 覆盖单位筛选完成
  end
  $U \leftarrow U_{avail}$
end
$U \leftarrow \{u_i \in U : u_i^{level} < u_j^{level}, \forall i,j \in \{1,\cdots,|U|\}$ 且 $i<j\}$
for $i=1,2,\cdots,|U|$ do
  img ← img $\cup\, u_i^{img}$ // 图像绘制
  if $u_i^{\text{cls}} \in C$ then
    box ← box $\cup\, u_i^{box}$ // 边界框保存
  end
end

算法 0-1 生成算法伪代码

完成绘制单位加入后，可以按照插入顺序得到待绘制单位序列 U，但生成的切片可能存在覆盖关系，因此需要引入最大覆盖率阈值 α，当被覆盖单位面积超过该单位切片面积的 α 倍时，对被覆盖单位进行去除，对单位完成筛选之后，再按照图层等级的从低到高进行绘制，并将识别类别 C 中的边界框信息进行记录，用于后续识别模型训练，具体绘制流程见算法 0-1。通过调整不同的单位生成数量、切片生成类型，最大覆盖阈值 α，可以得到如图 0-4 所示的生成结果。

(a) 单位生成数量 20，小型切片类型，$\alpha = 0.5$ (b) 单位生成数量 20，大型切片类型，$\alpha = 0.5$ (c) 单位生成数量 40，大型切片类型，$\alpha = 0.8$

图 0-4 生成式数据集实例

0.4.2 目标识别损失函数设计

我们沿用 Joseph Redmon 等人 [8] 提出的 YOLO 模型损失函数。设当前目标识别模型输出的网格步长为 s，输入图像的大小为 $W \times H$，每个网格处所需的预测框数目为 B，第 i 行 j 列网格预测出的第 k 个预测框记为 $\left(\widehat{box}_{ijk}, \hat{c}_{ijk}, \hat{p}_{c_{ijk}}, \widehat{bel}_{ijk}\right)$，对应的目标框 $(box_{ijk}, c_{ijk}, bel_{ijk})$，则该步长下的损失函数如式 (0-1) 所示。

$$
\begin{aligned}
\mathcal{L}_s(\hat{y}, y) = \sum_{i=1}^{H/s} \sum_{j=1}^{W/s} \sum_{k=1}^{B} & \mathbb{1}_{ijk}^{noobj} \lambda_{noobj} \mathcal{L}_{\text{BCE}}(\hat{c}_{ijk}, 0) \\
& + \mathbb{1}_{ijk}^{obj} \Bigg[\lambda_{box} \mathcal{L}_{\text{CIOU}} \left(\widehat{\text{box}}_{ijk}, \text{box}_{ijk} \right) + \lambda_{obj} \mathcal{L}_{\text{BCE}} \left(\hat{c}_{ijk}, \text{IOU}_{pred}^{true} \right) \\
& \quad + \lambda_{class} \sum_{c=1}^{C} \mathcal{L}_{\text{BCE}} \Big(\{\hat{p}_{ijk}\}_c, \text{onehot}\left(c_{ijk}\right)_c \Big) \\
& \quad + \lambda_{class} \mathcal{L}_{\text{BCE}} \left(\widehat{bel}_{ijk}, bel_{ijk} \right) \Bigg]
\end{aligned}
\tag{0-1}
$$

其中 $\mathbb{1}_{ijk}^{noobj}$ 当网格 (i, j) 下的第 k 个预测框没有对应的目标框时为 1，反之为 0，相反的 $\mathbb{1}_{ijk}^{obj} = 1 - \mathbb{1}_{ijk}^{noobj}$，$\mathcal{L}_{\text{CIOU}}$ 为 Zheng 等人 [9] 提出的 Complete-IOU 损失，$\mathcal{L}_{\text{BCE}}$ 为二元交叉熵损失，

$\lambda_{noobj}, \lambda_{box}, \lambda_{obj}, \lambda_{class}$ 分别为无标签，边界框位置信息，置信度以及类别对应损失的加权系数。

0.5 决策模型

0.5.1 特征设计

0.5.1.1 状态

模型的状态输入由 2 部分构成，分别为 $S^{img}, \boldsymbol{s}^{card}$，其中 $S^{img} \in \mathbb{R}^{18\times32\times15}$ 为单位的网格状特征输入，对于第 i 行 j 列的特征 $\boldsymbol{z}_{ij} := (S^{img})_{ij} \in \mathbb{R}^{15}$ 表示处于该位置的单位具有如下 4 种特征：$(\boldsymbol{z}_{ij})_{1:8}$ 为类别编码，$(\boldsymbol{z}_{ij})_9$ 为从属派别编码，$(\boldsymbol{z}_{ij})_{10:12}$ 为生命值图像编码，$(\boldsymbol{z}_{ij})_{13:15}$ 为其余条状图像编码；$\boldsymbol{s}^{card} \in \mathbb{R}^6$ 表示当前状态下的两个全局特征：$(\boldsymbol{s}^{card})_{1:5}$ 为当前手牌信息，$(\boldsymbol{s}^{card})_6$ 为当前总圣水量。

0.5.1.2 动作

模型的动作输入由 2 个部分构成：$\boldsymbol{a}^{pos}, a^{select}$，其中 $\boldsymbol{a}^{pos} \in \mathbb{R}^2$ 表示动作执行的部署坐标，a^{select} 表示动作执行的手牌编号。

0.5.1.3 奖励

奖励设计如下，设 $h_i^{\text{bel}}, (i \in \{0,1,2\}, \text{bel} \in \{0,1\})$ 为防御塔生命值，当 $i = 1,2$ 时表示左右两个副塔生命值，$i = 0$ 表示主塔生命值，$\text{bel} = 0,1$ 分别表示我方和敌方建筑，Δh_i^{bel} 表示前一帧与当前帧生命值的差值，H_i^{bel} 表示对应防御塔的总生命值，分别定义如下四种奖励函数：

1. 防御塔生命值奖励

$$r_{tower} = \sum_{\text{bel}=0}^{1}\sum_{i=0}^{2}(-1)^{\text{bel}+1}\frac{\Delta h_i^{\text{bel}}}{H_i^{\text{bel}}} \tag{0-2}$$

2. 防御塔摧毁奖励 $r_{distory}$：当敌我副塔被摧毁时给予 $(-1)^{\text{bel}+1}$ 奖励，敌我主塔被摧毁时给予前者的 3 倍奖励。

3. 主塔激活奖励 $r_{activate}$：当副塔均存活的条件下，主塔第一次失去生命值时，给予 $(-1)^{\text{bel}}$ 0.1 奖励。

4. 圣水溢出惩罚 r_{elixir}：当总圣水持续保持溢出状态时，每间隔 1 秒产生一次 0.05 的惩罚。

综合上述奖励，得到总奖励

$$r = r_{tower} + r_{destory} + r_{activate} + r_{elixir} \tag{0-3}$$

0.5.2 离线强化学习

首先对强化学习中概念进行介绍，考虑无限长带折扣 Markov 决策过程（Markov Decision Process, MDP），定义为 $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$，其中 $\mathcal{S}$ 为状态空间，$\mathcal{A}$ 为动作空间，$p : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ 为状态转移方程，$r : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ 为奖励函数，$\gamma \in (0,1)$ 为折扣系数。令 π 表示决策函数 $\pi : \mathcal{S}\times\mathcal{A} \to [0,1]$，令 $R(\pi)$ 表示其期望所获得的总奖励（回报）：

$$R(\pi) = \mathbb{E}_{S_1,A_1,S_2,A_2\cdots}\left[\sum_{t=0}^{\infty} r(S_t, A_t)\right], \quad \text{其中 } A_t \sim \pi(\cdot|S_t), S_{t+1} \sim p(\cdot|s_t, a_t) \tag{0-4}$$

强化学习的目标通常是找到最优策略 $\pi^* := \arg\max_\pi R(\pi)$，在线强化学习算法往往通过策略迭代和价值函数估计方法实现策略的更新，而本文中所使用的离线强化学习算法不再基于值估计方法，而是更加类似于模仿学习的方法。

Chen 等人 [7] 的 Decision Transformer（DT）是一种将强化学习问题是为序列建模问题的方法，使用了深度学习中的 Transformer 架构，对于离线数据集中的一段长度为 T 的交互轨迹（Trajectory）

$$\rho = (s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T, s_{T+1}) \tag{0-5}$$

其中 s_{T+1} 为终止状态，则 ρ 可以视为建模为序列

$$R_0, s_1, a_1, R_1, s_2, \cdots, a_{T-1}, R_{T-1}, s_T, a_T \tag{0-6}$$

其中 $R_i = \sum_{t=i}^{T} r_{t+1}, (i = 0, \cdots, T-1)$ 为目标回报（Return-to-Go）。

DT 模型中序列编码模型使用的是 Radford 等人 [10] 的 GPT 模型，即仅含有编码器的因果注意力机制。因果注意力（Causal Attention）为（每个特征 i 只能看到 $j \leqslant i$ 的特征）

$$Z = \text{softmax}\left(\frac{(QK^T) \odot M}{\sqrt{d_k}}\right) V \tag{0-7}$$

其中 M 被成为掩码矩阵，为 N 阶下三角阵，Q, K, V 分别表示输入序列 X 对应生成的询问键（Query），查询键（Key）和价值键（Value），询问键与查询键具有相同的特征维度 d_k。当式 (0-7) 中 M 为全 1 矩阵时，Z 定义为交叉注意力（Cross Attention）。

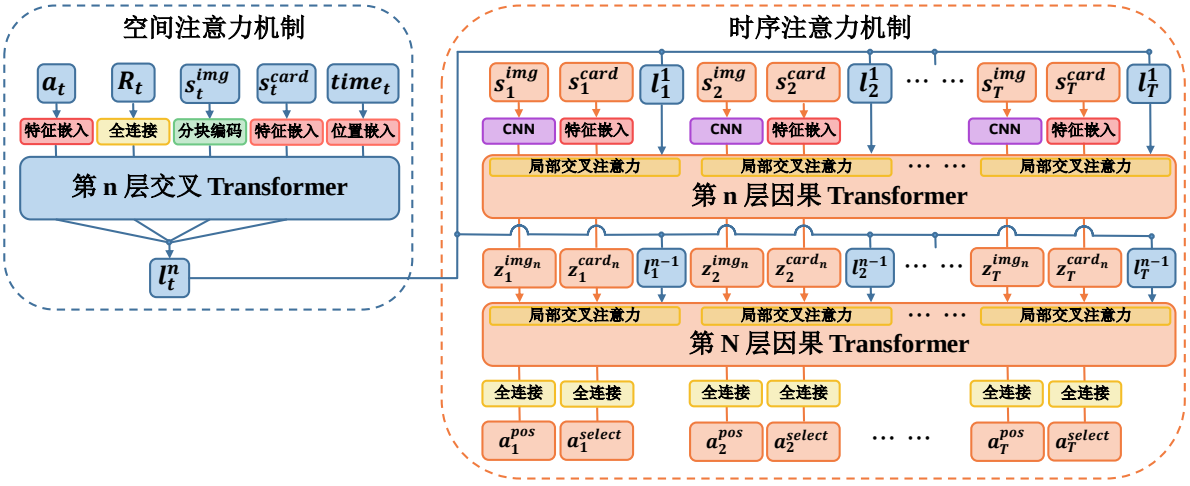


图 0-5 决策模型架构，左侧的空间注意力机制用于编码同一时刻下的特征信息，右侧的时序注意力机制用于关联上下帧信息，做出动作预测

我们设计的模型如图 0-5 所示，基于 Shang 等人的 [11]StARformer 的 ViT+DT 的架构，模型输入为轨迹序列 $(a_t, R_t, s_t)_{t=1}^T$，输出为动作预测序列 $(a_t^{pos}, a_t^{select})_{t=1}^T$。左侧交叉注意力机制对局部信息 (a_t, R_t, s_t) 按空间维度进行编码，并使用 ViT 中分块思路将图像 s_t^{img} 转化为序列；右侧为因果注意力机制对全局信息 (s_t^{img}, s_t^{card}) 按时序维度进行编码，并在每层序列输入中引入对应的局部编码信息 l_t^n。由于需要使同一时刻下的信息可以相互产生注意力关系，所以需要引入局部交叉注意力，具体实现方法是将式 (0-7) 中的掩码矩阵 M 定义为 M_3，其中

$$(M_{L_0})_{ij} = \begin{cases} 1, & i = kL_0 - l, j \leqslant kL_0 \\ 0, & \text{否则} \end{cases}, \quad k \in \{1, \cdots, L\}, l \in \{0, \cdots, L_0 - 1\} \tag{0-8}$$

设输入的轨迹的长度为 L，$\mathbb{H}$ 表示占位符，我们设计了三种模型架构：

- StARformer-3L 模型输入序列长度为 $3L$，第 n 层 Cause Attention Transformer 输出的时序序列记为 $\{z_t^{img_n}, z_t^{card_n}, l_t^{n-1}\}_{t=1}^{L}$，局部注意力中掩码矩阵为 M_3，输出与 $(a_t^{pos}, a_t^{select}, \mathbb{H})_{t=1}^{T}$ 对应。
- StARformer-2L 模型输入序列长度为 $2L$，第 n 层 Cause Attention Transformer 输出的时序序列记为 $\{z_t, l_t^{n-1}\}_{t=1}^{L}$，局部注意力中掩码矩阵为 M_2，输出与 $([a_t^{pos}, a_t^{select}], \mathbb{H})_{t=1}^{T}$ 对应。
- DT-4L 模型输入长度为 $4L$，仅包含时序注意力机制中的 Cause Attention Transformer，第 n 层 Transformer 的输出的时序序列为 $\{a_{t-1}, R_{t-1}, s_t^{img_n}, s_t^{card_n}\}_{t=1}^{L}$，输出与 $(\mathbb{H}, \mathbb{H}, a_t^{pos}, a_t^{select})_{t=1}^{T}$ 对应。

0.5.3 预测目标设计与重采样

由于本任务中动作执行极为离散，总帧数中仅有 4% 为执行动作帧，其余帧均不执行动作，如果直接逐帧预测动作会产生非常严重的长尾问题，导致模型最终基本不执行动作（表 0-3 中离散预测的动作数远低于连续动作预测数），因此需要将预测目标从离散转化为连续，解决方法是引入延迟动作预测：对于第 i 帧，需找到其后（包含自身）最近的动作帧 j，令最大间隔帧数阈值为 T_{delay}，则每个非动作帧的预测的延迟动作为 $a_i^{delay} = \min\{j - i, T_{delay}\}$。

对离线数据集进行采样时，为避免长尾问题导致模型偏移，本文还设置了重采样频次，设数据集总帧数为 N，动作帧数为 N_{action}，则动作帧占比为 $r_a := N_{action}/N$，对于第 i 个动作帧位于数据集中的第 t_i 帧，则轨迹的结束帧 $j \in \{t_i, \cdots, t_{i+1} - 1\}$ 对应的重采样频次为

$$s_j = \max\left\{\frac{1}{1 - r_a}, \frac{1}{r_a(j - t_i + 1)}\right\}, \quad (t_i \leqslant j \leqslant t_{i+1}) \tag{0-9}$$

则训练轨迹中结束帧的采样分布为 $\left\{\frac{s_j}{\sum_{j=1}^{N} s_j}\right\}_{j=1}^{N}$，图 0-6 中展示了离线数据集一段轨迹所对应的重采样频次与动作预测值。

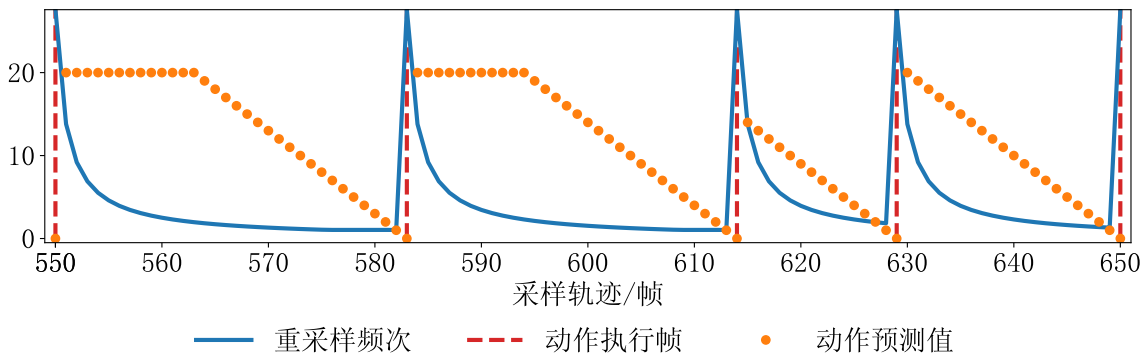


图 0-6 从离线数据集中截取的一段数据，总共包含 5 个动作帧，最大间隔帧数阈值 $T_{delay} = 20$，

0.6 数据分析及实验结果

本章对前 3 章内容的实验结果进行总结，分别包含第 0.4 章的生成式数据集分析统计和目标识别模型对比实验，以及 0.6.3 的决策模型对比实验。

0.6.1 生成式数据集分析

数据集总共分为两部分[③]：

1. 生成式数据集切片：总计 154 个类别，待识别类别 150 个，总共包含 4654 个切片，在全部待识别类别的切片图像中，切片大小分布如图 0-7 所示。
2. 目标识别验证集：总计 6939 张人工标记的目标识别图像，包含 116878 个目标框，平均每张图片包含 17 个目标框，该数据集均为真实对局视频流逐帧标记得到，而模型训练所使用的完全是生成式数据集，所以该数据集可以做验证集使用。

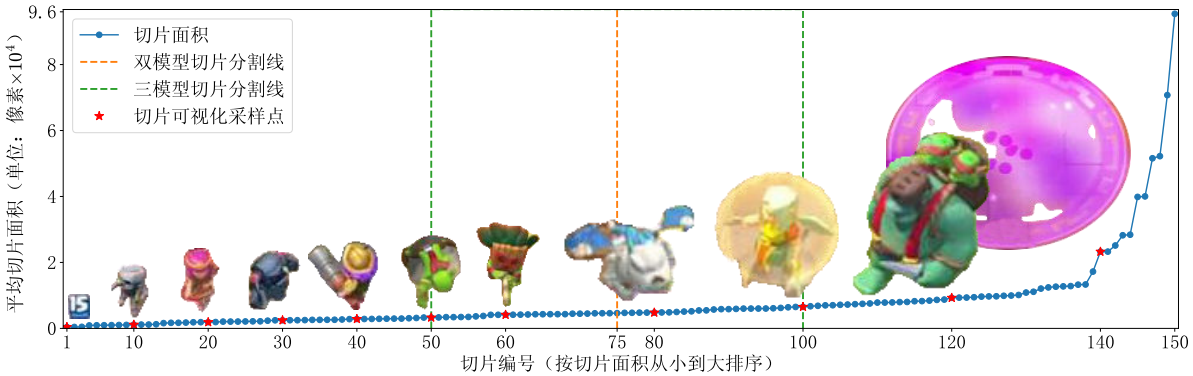


图 0-7 将切片数据集全部切片按平均面积从小到大排序并编号，从中随机采样出部分切片进行可视化。分别以全体切片面积的二等分和三等分点，作为双模型和三模型的识别类别分割线。

0.6.2 目标识别模型

目标识别模型使用了自己实现的 YOLOv5[④] 和重构后的 YOLOv8 的模型[⑤]，每个训练集大小设置为 20000，至多训练 80 个 epoch 收敛。数据增强使用了：HSV 增强，图像旋转，横纵向随机平移，图像缩放，图像左右反转，具体参数见附录表 ?? 。

实验结果[⑥]如表 0-2 所示，表中具体内容解释如下：

1. 模型名称：编号后的字母表示模型大小，l,x 分别对应大与特大型模型，YOLOv8-l×n 表示使用 n 个 YOLOv8-l 模型，每个子模型分别识别图 0-7 中分割线所划分区域中的切片类型，最后将识别的预测框通过非最大值抑制（Non-Maximum Suppression，NMS）进行筛选，NMS 过程中 IOU 阈值设定为 0.6。

2. 评测指标：表中 mAP 评测指标表示 COCO mAP 指标[?]，即在 10 种不同 IOU 阈值下计

[③]上述数据集统计信息截止于 2024 年 5 月 6 日，全部图像数据集均已开源：https://github.com/wty-yy/Clash-Royale-Detection-Dataset

[④]复现 YOLOv5 代码：https://github.com/wty-yy/KataCV/tree/master/katacv/yolov5

[⑤]YOLOv8 重构内容：https://github.com/wty-yy/KataCR/blob/master/asserts/yolov8_modify.md

[⑥]YOLOv5 全部训练曲线：https://wandb.ai/wty-yy/ClashRoyale，YOLOv8 全部训练曲线：https://wandb.ai/wty-yy/YOLOv8

算 PR 曲线下面积求平均得到，AP50、P50、R50 和 mAP(S) 分别表示在判断正例的 IOU 阈值为 50% 下的 mAP、平均精度、平均召回率和小目标的 mAP。

3. 验证速度：模型预测时 Batch 大小设置为 1，FPS 为模型在 GeForce RTX 4090 下测试的验证速度，验证测试时置信度设置为 0.001。当对视频流数据进行预测时，将置信度改为 0.1，并使用 ByteTrack[?] 算法在目标追踪计算过程中对边界框进行筛选，FPS(T) 是在 GeForce RTX 4060 Laptop 下带有目标追踪的识别速度。

从实验结果可以看出，YOLOv8-l 的双识别器对小目标的识别能力与三识别器效果基本一致，并远高出非组合式的识别器，其原因可能在于 150 个预测类别大小远超模型的识别能力范围，最大目标与最小目标的边界框大小差距甚远，又由于 YOLOv8 是无锚框识别头，由于大目标易于识别，可能导致预测的目标框均偏大，所以多个识别器降低平均类别数能够有效对小目标进行识别。

表 0-2 YOLO 模型对比测试结果

模型名称	AP50	P50	R50	mAP	mAP(S)	FPS	FPS(T)	检测器类别数	数据增强
YOLOv5-l	66.2	84.4	63.8	53.2	NA	59	NA	151	
YOLOv8-x	83.1	**93.9**	68.3	67.7	39.8	68	31	160	
YOLOv8-x	**85.3**	90.7	80.4	66.8	35.9	68	31	160	✓
YOLOv8-l ×2	84.3	89.5	79.8	67.4	43.9	34	18	85	✓
YOLOv8-l ×3	85.2	89.7	**80.9**	**68.8**	**48.3**	23	10	65	✓

0.6.3 决策模型

我们手动构建了玩家与与游戏内置的 8000 分 AI 对局 105 回合的专家数据[⑦]，固定双方使用的卡组，数据集总计 113981 帧，动作帧占比 4.01%，重采样频次比率为 $\frac{\text{动作帧}}{\text{非动作帧}} = 24.92 : 1.04$，平均动作延迟大小为 21.26，最大间隔帧数阈值为 $T_{delay} = 20$（重采样细节见 ??）。模型损失函数分为三个部分，由于均为离散动作，所以损失函数均使用目标动作均使用交叉熵损失，总损失函数如下

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{I}_{a_i^{delay} < T_{delay}} \left[\mathcal{L}_{\text{CE}}(\hat{\boldsymbol{a}}_i^{pos}, \boldsymbol{a}_i^{pos}) + \mathcal{L}_{\text{CE}}(\hat{a}_i^{select}, a_i^{select}) + \mathcal{L}_{\text{CE}}(\hat{a}_i^{delay}, a_i^{delay}) \right] \tag{0-10}$$

其中 $T_{delay}, \boldsymbol{a}_i^{pos}, a_i^{select}, a_i^{delay}$ 分别为最大间隔帧数阈值、目标动作的部署坐标、手牌编号以及部署延迟，注意每条轨迹下只考虑 $a_i^{delay} < T_{delay}$ 对应的梯度。

本文分别测试了下述模型参数：

- 不同的模型架构见 0.5.2 ，分别测试了 StARformer 和 DT 架构。
- 模型输入的轨迹步长记为 L，测试了 $L = 30, 50, 100$ 的情况。
- 不同的预测目标（离散与连续预测见 0.5.3 ）。
- 不同的手牌预测范围，默认预测当前手牌编号，也尝试了对当前牌库中全部手牌进行预测。

本文使用了如下数据增强方式对模型进行训练：

[⑦]专家数据集：`https://github.com/wty-yy/Clash-Royale-Replay-Dataset`

- 重采样：对稀疏的动作帧按比例进行进行重采样，加快模型收敛，缓解离线数据集的长尾问题。
- 随机手牌重组：对当前输入轨迹中的全部手牌按照随机排列进行打乱，当预测全部手牌编号时，将动作对应的手牌也进行相应变换。

全部模型训练曲线均进行了上传[8]，模型实时对局的验证结果总结于表 0-3 中，在实时对局的实现中包含以下细节：

- 动作执行：对于连续动作预测模型中，由于感知识别中存在延迟，当预测动作延迟在 8 帧以内就会立刻执行动作，并且为了避免总圣水溢出导致的惩罚奖励，每当总圣水达到 10 时就直接下出当前预测的卡牌。
- 无效动作跳过：若模型预测出的卡牌所需圣水超出了当前总圣水量或当前动作执行的卡牌位为空。

表 0-3 决策模型对比

模型框架	步长 L	训练回合	总奖励	对局时长/秒	动作数	胜率
DT-4L	50	8	-5.7 ± 2.5	148.9 ± 33.6	128.7 ± 37.7	5%
StARformer-2L	30	3	-6.0 ± 2.3	135.0 ± 35.1	141.8 ± 57.9	5%
StARformer-2L	50	8	-6.2 ± 2.2	131.9 ± 44.3	195.3 ± 69.8	5%
StARformer-2L	100	1	-4.9 ± 2.8	150.2 ± 35.6	187.6 ± 48.2	0%
StARformer-3L	30	4	-5.1 ± 3.7	147.2 ± 37.4	190.8 ± 52.7	**10%**
StARformer-3L	50	3	$\mathbf{-4.7 \pm 3.1}$	$\mathbf{158.9 \pm 27.7}$	$\mathbf{207.8 \pm 48.2}$	5%
StARformer-3L	100	5	-6.1 ± 2.2	125.9 ± 37.8	144.6 ± 42.9	5%
StARformer-3L（全卡牌预测）	50	2	-5.6 ± 2.1	150.2 ± 38.6	195.3 ± 69.8	0%
StARformer-2L（离散动作预测）	50	1	-7.5 ± 0.8	123.1 ± 39.2	21.9 ± 9.4	0%

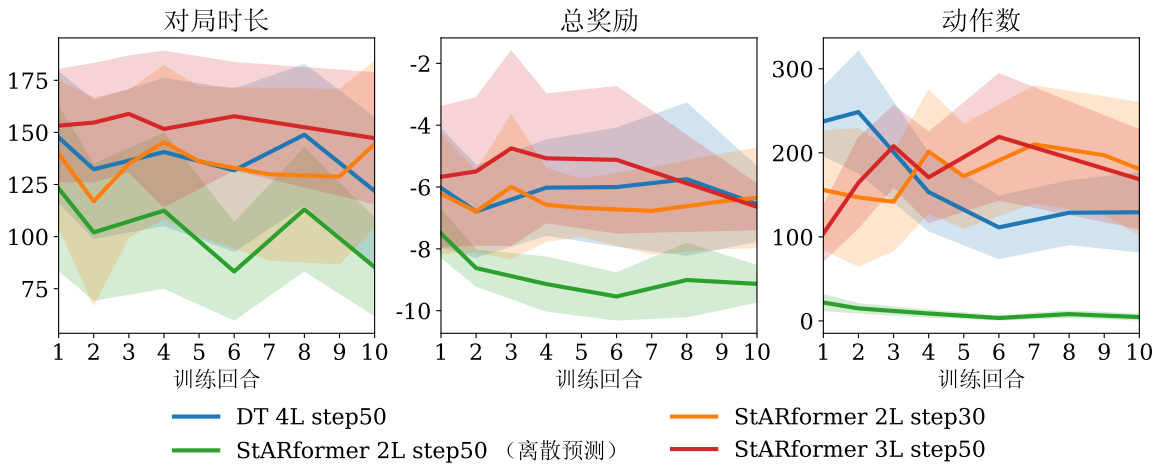
表 0-3 中记录了为每个模型的前 10 次训练结果中，与环境交互 20 个回合得到的最高平均奖励，从中可以看出，将离散预测改为连续预测提高了 37% 的性能、StARformer 架构从 2L 修改为 3L 的改动提高了 24% 的模型性能。表中每列的含义分别为：

- 步长 L：为模型架构设计 0.5.2 中的输入轨迹长度。
- 训练回合：前 10 个训练结果中，获得最高奖励所对应的回合数。
- 总奖励：按奖励公式 (0-3) 进行累计得到的总奖励。
- 对局时长：统计每次对局的时长。
- 动作数：统计每局智能体成功执行的动作数目。
- 胜率：按照 0.3.2.0.2 中介绍的游戏目标，判定智能体的获胜概率。

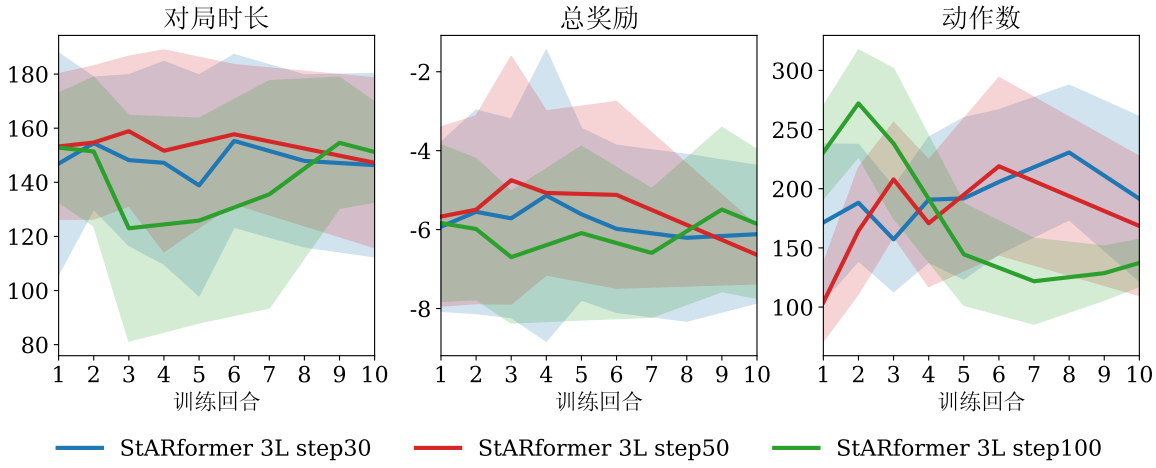
本文使用的验证环境：手机系统为鸿蒙、电脑系统为 Ubuntu24.04 LTS、CPU: R9 7940H、GPU: RTX GeForce 4060 Laptop，平均决策用时 120ms，感知融合用时 240ms。

图 0-8 中展示了每种模型前 10 个训练结果的验证曲线。

[8]决策模型训练曲线：`https://wandb.ai/wty-yy/ClashRoyale%20Policy`



(a) 不同模型结构



(b) StARformer-3L 采取不同轨迹长度

图 0-8 模型验证曲线：展示了前 10 个训练结果，每回合模型在真实对局中的对局时间时长、总奖励和执行动作数，每次进行 20 次对局，实线为均值、虚影为标准差。

0.7 总结

本文基于游戏皇室战争（Clash Royale），首次提出了一种基于非嵌入式的离线强化学习训练策略。结合目标识别和光学文本识别的前沿算法，成功实现了智能体在移动设备上进行实时对局，并且战胜了游戏中的内置 AI。

我们为非嵌入式强化学习在移动设备上的应用提供了新的思路。未来的工作可以从算法改进上进行拓展，当前在固定卡组下进行训练，仍然无法 100% 战胜游戏中的内置 AI，因此远无法达到人类平均水平，而且制作离线强化学习数据集需要花费大量的人力，若要进一步提升智能体能力，应该需要采用在线强化学习算法，与此同时需要使用更加高效的感知融合算法和决策模型架构，才有可能进一步提高智能体的实时决策能力和对局胜率。

本文的全部代码均已开源[9]，智能体对局获胜的视频已上传[10]，期望能够为相关领域的研究者提供有价值的参考和借鉴。

0.8 致谢

[9]全部代码：https://github.com/wty-yy/katacr

[10]对局视频：https://www.bilibili.com/video/BV1xn4y1R7GQ

参考文献

[1] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540):529-533.

[2] Silver D, Huang A, Maddison C J, et al. Mastering the game of go with deep neural networks and tree search[J]. nature, 2016, 529(7587):484-489.

[3] Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning[J]. Nature, 2019, 575(7782):350-354.

[4] Berner C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning[J]. arXiv preprint arXiv:1912.06680, 2019.

[5] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. arXiv preprint arXiv:1707.06347, 2017.

[6] Kumar A, Zhou A, Tucker G, et al. Conservative q-learning for offline reinforcement learning[J]. Advances in Neural Information Processing Systems, 2020, 33:1179-1191.

[7] Chen L, Lu K, Rajeswaran A, et al. Decision transformer: Reinforcement learning via sequence modeling[J]. Advances in neural information processing systems, 2021, 34:15084-15097.

[8] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[9] Zheng Z, Wang P, Liu W, et al. Distance-iou loss: Faster and better learning for bounding box regression [C]//Proceedings of the AAAI conference on artificial intelligence: volume 34. 2020: 12993-13000.

[10] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. 2018.

[11] Shang J, Kahatapitiya K, Li X, et al. Starformer: Transformer with state-action-reward representations for visual reinforcement learning[C]//European conference on computer vision. Springer, 2022: 462-479.