



## 本科毕业设计（论文）

基于计算机视觉及强化学习的非嵌入式游戏 AI 设计

学院（部、中心）：数学与统计学院

专业：应用数学系

班级：强基数学 002

学生姓名：吴天阳

学号：2204210460

指导教师：康艳梅，兰旭光

2024 年 5 月

## 摘要

本文首次提出了一种基于非嵌入式离线强化学习的训练策略，应用于游戏皇室战争（Clash Royale）。结合当前目标识别与光学文本识别的顶尖算法，使用离线强化学习算法制定策略，成功实现了智能体在非嵌入条件下的实时对局：移动设备上实时图像获取、感知融合、智能模型决策及控制移动设备执行动作，从而能够与对手进行实时对局，并且能够战胜游戏中的内置 AI。

首先，本文设计了一种高效制作切片数据集的方法，基于该方法制作了包含 4654 张切片、共 150 个类别的切片数据集，以及包含 116878 个目标框、共 6939 张图像的目标识别数据集<sup>①</sup>，并提出了一种可行的生成式目标识别数据集算法，使其能够模拟真实对局场景生成含有任意数量部队及种类的带标签图像。通过生成式图像训练出的模型在真实视频流中表现出良好的泛化性，具有很高的识别准确率。

其次，本文基于计算机视觉模型的输出结果设计了感知融合算法，该算法结合视频数据中的上下文信息优化特征结果，从而进一步提升识别的准确率。

最后，在决策模型方面，本文从架构及预测目标两个方面对传统模型进行了改进，将难以学习的离散动作序列转化为连续动作序列，大幅提高了模型性能。本文制作了包含 105 回合、总共 113981 帧的专家数据集<sup>②</sup>，并基于该离线数据集在不与真实环境交互的条件下，训练出了能够战胜游戏内置 AI 的智能体。

本文的全部代码均已开源<sup>③</sup>。

**关键词：**目标识别；光学文本识别；强化学习

---

<sup>①</sup> 图像数据集：<https://github.com/wty-yy/Clash-Royale-Detection-Dataset>

<sup>②</sup> 专家数据集：<https://github.com/wty-yy/Clash-Royale-Replay-Dataset>

<sup>③</sup> 全部代码：<https://github.com/wty-yy/katacr>

## ABSTRACT

This paper introduces, for the first time, a non-embedded offline reinforcement learning training strategy applied to the game Clash Royale. By combining state-of-the-art algorithms in object recognition and optical character recognition, we use an offline reinforcement learning algorithm to develop strategies, successfully enabling real-time gameplay by the agent under non-embedded conditions: real-time image capture on a mobile device, perception fusion, intelligent model decision-making, and control of the mobile device to execute actions. As a result, the agent can engage in real-time matches against opponents and defeat the game's built-in AI.

First, we designed an efficient method for creating slice datasets. Using this method, we created a slice dataset containing 4654 slices across 150 categories, and an object recognition dataset containing 116878 target boxes across 6939 images<sup>④</sup>. We also proposed a feasible generative algorithm for creating object recognition datasets, allowing the generation of labeled images with any number and type of units that mimic real match scenarios. Models trained on these generative images demonstrated excellent generalization performance in real video streams, achieving high recognition accuracy.

Additionally, we designed a perception fusion algorithm based on the output results of computer vision models, which optimizes feature results by incorporating contextual information from video data, further improving recognition accuracy.

Finally, in terms of decision-making models, we improved traditional models from both the architecture and prediction target perspectives. By transforming difficult-to-learn discrete action sequences into continuous action sequences, this method significantly enhanced model performance. We created an expert dataset containing 105 rounds and a total of 113981 frames<sup>⑤</sup>. Based on this offline dataset, the agent was trained to defeat the built-in AI without interacting with the real environment.

All the code used in this paper has been open-sourced<sup>⑥</sup>.

KEY WORDS: Object Recognition; Optical Character Recognition; Reinforcement Learning

---

<sup>④</sup> Image dataset: <https://github.com/wty-yy/Clash-Royale-Detection-Dataset>

<sup>⑤</sup> Expert dataset: <https://github.com/wty-yy/Clash-Royale-Replay-Dataset>

<sup>⑥</sup> All code: <https://github.com/wty-yy/katacr>

## 目 录

摘 要.....	I
ABSTRACT .....	II
主要符号表.....	V
1 绪论.....	1
1.1 嵌入式智能体研究现状.....	1
1.2 计算机视觉研究现状 .....	2
1.2.1 图像分类.....	2
1.2.2 目标识别 .....	2
1.2.3 光学字符识别 .....	2
1.3 本毕设的任务 .....	3
1.4 本毕设的贡献 .....	4
2 目标识别数据集搭建及模型架构设计.....	5
2.1 切片数据集制作流程 .....	5
2.2 目标识别模型设计 .....	6
2.3 生成式目标识别数据集.....	8
3 感知融合 .....	11
3.1 感知模型.....	11
3.1.1 目标识别模型 .....	11
3.1.2 光学字符识别模型.....	11
3.1.3 图像分类模型 .....	12
3.2 感知特征提取 .....	12
3.2.1 状态特征提取 .....	12
3.2.2 动作特征提取 .....	14
3.2.3 奖励特征提取 .....	14
4 决策模型.....	16
4.1 离线强化学习 .....	16
4.1.1 Decision Transformer .....	16
4.1.2 StARformer.....	17

4.2 决策模型设计 .....	18
4.2.1 状态空间与动作空间 .....	18
4.2.2 预测目标设计与重采样 .....	18
4.2.3 模型架构设计 .....	19
5 数据分析及实验结果 .....	21
5.1 生成式数据集分析 .....	21
5.2 目标识别模型训练 .....	21
5.3 决策模型训练 .....	22
6 总结与展望 .....	25
致 谢 .....	26
参考文献 .....	27
附录 A 相关内容补充 .....	30
A.1 动态概率分布 .....	30
A.2 数据增强 .....	30
A.3 模型测试卡组 .....	31
附录 B 论文相关代码 .....	32
B.1 生成式数据集 .....	32
B.2 特征融合 .....	33
B.2.1 状态特征提取部分代码 .....	33
B.2.2 动作特征提取部分代码 .....	34
B.2.3 奖励特征提取部分代码 .....	34

## 主要符号表

$\mathbb{R}$	实数集
$\mathbb{N}$	自然数集
$\mathbb{Z}$	整数集
$\mathbb{Z}^+$	正整数集
$\mathbb{R}^N$	$N$ 维空间, 由 $N$ 个实数集 $\mathbb{R}$ 所构成的笛卡尔积
[条件]	当“条件”成立时为 1, 否则为 0
$A_{ij}$	矩阵 $A$ 的第 $i$ 行第 $j$ 列元素
$x_i$	向量 $\mathbf{x}$ 中第 $i$ 个元素
$\mathbf{x}_{i:j}$	向量 $\mathbf{x}$ 中第 $i$ 到 $j$ 个元素所构成的向量
$\text{softmax}(\mathbf{z})$	$\left\{ \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}} \right\}_{i=1}^C$ 其中向量 $\mathbf{z} \in \mathbb{R}^C$ , $C$ 为总类别数
$A \odot B$	$(A \odot B)_{ij} = (A)_{ij}(B)_{ij}$ 其中矩阵 $A$ 与矩阵 $B$ 具有相同形状

# 1 绪论

## 1.1 嵌入式智能体研究现状

当前强化学习（Reinforcement Learning, RL）已经成为人工领域备受关注的重点之一，它通过智能体与环境交互利用得到的奖励大小来指导策略的改进，通过不断地试错来获得最优策略。

强化学习在游戏领域中有着广泛的应用：围棋作为经典博弈游戏，因其具有巨大的状态空间以及复杂的策略而无法使用传统搜索算法解决，DeepMind 团队通过 AlphaGo<sup>[1]</sup>给出了基于价值函数估计和蒙特卡洛树搜索的解决方法，首次打败人类顶级选手，标志着强化学习在处理复杂游戏中的巨大成功；AlphaZero<sup>[2]</sup>也使用类似模型架构，但它完全基于自我博弈学习，在摒弃任何人类先验知识的前提下超越了 AlphaGo，它展现了自博弈学习的强大能力。

类 MOBA 游戏（多人在线战术竞技游戏）中强化学习也有着非凡的成就，这类游戏往往有着巨大的动作、状态空间，并带有高动态的前后文信息，需要智能体之间的团队配合能力。OpenAI 公司通过 OpenAI Five<sup>[3]</sup>给出了基于强化学习的解决方法，该深度网络模型为长短期记忆循环神经网络（Long Short-Term Memory, LSTM）<sup>[4]</sup> 和卷积神经网络（Convolutional Neural Network, CNN）组合，提出并利用了强化学习中经典在线同策略近端策略优化算法（Proximal Policy Optimization, PPO）算法<sup>[5]</sup>，使用分布式训练方法使其首次在 Dota2 游戏中战胜顶级职业选手。类似的还有 DeepMind 团队的 AlphaStar<sup>[6]</sup> 在即时战略游戏《星际争霸》中解决了复杂的多智能体决策问题。

值得注意的是上述模型使用的算法均为**在线强化学习算法**即经典强化学习研究内容，经典算法例如 PPO<sup>[5]</sup>、深度 Q 函数学习（Deep Q Network, DQN）<sup>[7]</sup>、软演员评论家算法（Soft Actor-Critic, SAC）等，这是因为**嵌入式智能体**与游戏环境之间本身具有低成本高交互次数的特性，使得智能体有无限的试错空间，可以不断对新状态空间进行探索，更好地对动作状态价值函数  $Q(s, a)$  进行估计，并进一步利用  $Q(s, a)$  对自身策略  $\pi(a|s)$  进行优化，从而通过迭代方法接近最优策略  $\pi^*(a|s)$ 。

由于嵌入式智能体对环境信息的读取和人类完全不同，人类理解视频游戏的主要方式是通过图像信息，而**嵌入式智能体可以直接获取精确的游戏内部信息**，虽然这些人类也可以通过图像信息间接获得，但可视化的图像信息中往往伴有噪声，在信息输入方面就会产生误差，因此就导致了游戏竞技上的不公平性产生（其余的不公平因素，例如感知频率与操作频率，对智能加入相应限制可以解决）。

由于当前计算机视觉技术（Computer Vision, CV）发展迅速，本论文提出一种非嵌入智能体的特征提取方法，这种**非嵌入式智能体可以仅通过获取和人类完全相同的图像信息**进行决策，由于非嵌入式智能体只能通过模拟器和环境进行交互，无法高效获取数据，所以本文只能收集专家数据集，采用 4.1 中所介绍的离线强化学习算法 Decision

Transformer (DT)<sup>[8]</sup> 和 StARformer<sup>[9]</sup> 来完成决策部分。

## 1.2 计算机视觉研究现状

非嵌入式智能体主要利用到计算机视觉中图像分类 (Image Classification)、目标识别 (Object Detection) 和光学字符识别 (Optical Character Recognition, OCR) 三个方面，下面分别对其进行简单介绍：

### 1.2.1 图像分类

在图像分类任务上 LeNet-5<sup>[10]</sup> 给出了早期卷积神经网络 (Convolutional Neural Network, CNN) 模型，它作为现在 CNN 架构的先驱之一，在 MNIST 手写数据集上首次展现出深度神经网络的卓越性能。近年来由于自动微分计算速度的飞速提升以及 ImageNet2012 数据集<sup>[11]</sup> 的支撑，从而使得基于数据驱动的模型重新被重视，自从 AlexNet<sup>[12]</sup> 利用在 2012 年 ImageNet 挑战赛中取得突破性成果后，各类新型网络架构例如 VGG<sup>[13]</sup>、GoogleNet<sup>[14]</sup>、ResNet<sup>[15]</sup>、DenseNet<sup>[16]</sup>、ViT<sup>[17]</sup>、RepVGG<sup>[18]</sup>、MetaFormer<sup>[19]</sup> 等相继被提出，这些模型不断刷新图像分类的准确率记录，同时提升了模型的计算能力和泛化能力。

研究者针对网络的宽深度 (VGG)、宽度 (GoogleNet)、残差连接 (ResNet)、分组卷积 (AlexNet)、注意力机制 (ViT, MetaFormer)、模型重新参数化 (RepVGG) 等方面进行了深入探索，其中残差网络的引入解决了深度网络的训练难题，可以将网络深度上升到数百层之上，乃至 Transformer<sup>[20]</sup> 类架构中也使用了残差机制；而注意力机制的引入可以帮助模型关注图像的关键特征。

### 1.2.2 目标识别

目标识别任务 (Object Detection) 旨在从图像或视频数据中识别出特征类别的目标或对象，根据处理流程的不同检测头 (Detection Head) 分为单阶段检测器 (Single-Stage Detectors) 和多阶段检测器 (Two-Stage Detectors)，其中单阶段检测器，通常为端到端问题，直接通过输入的图像给出识别框的位置 (回归问题) 与类别 (分类问题)，当前经典模型包括 YOLO (You Only Look Once) v1 ~ 9 系列<sup>[21]</sup>、SSD (Single Shot Multibox Detector)<sup>[22]</sup>、DETR (DEtection TRansformer)<sup>[23]</sup> 等。这类算法优点在于推理速度较快，适用于实时监测任务，该系列在初次提出时检测率较低，但通过不断的优化改进，已经可以成功的解决小目标检测和重叠目标问题。而多阶段检测器例如 R-CNN<sup>[24]</sup> 系列，这类算法虽精度较高，但推理速度较长，不适用于实时监测任务。

### 1.2.3 光学字符识别

光学字符识别 (OCR, Optical Character Recognition) 目前最常用的模型是卷积循环神经网络 (Convolutional Recurrent Neural Networks, CRNN)<sup>[25]</sup>，其将经典序列损失向

题损失函数 CTC (Connectionist Temporal Classification)<sup>[26]</sup> 与深度神经网络架构 LSTM 和 CNN 相结合，首先利用 CNN 将图像信息编码为一维序列信息，再用 (Bi-)LSTM<sup>[27]</sup> 进行文本预测，最后利用 CTC 损失函数使用对预测的文本顺序进行纠正。

但是仅有文字识别模型并不足够，还需要对图像中的文本位置进行定位再使用 CRNN 算法进行识别，于是发展出了很多 OCR 系统，如百度公司的 PP-OCRv1 ~ 4 系列<sup>[28]</sup>。

为了理解每个模型的搭建与训练细节，本毕设对上述介绍的部分模型进行了复现<sup>①</sup>。

### 1.3 本毕设的任务

《皇室战争》(Clash Royale, CR) 是一款卡牌类即时策略游戏 (RTS)，本文仅考虑一对一对战模式，双方玩家需要通过实时决策使用手牌来战胜对手，首先对游戏基本规则及非嵌入式感知任务目标进行介绍：

**感知场景** 在对局过程中，游戏场景如右图 1-1 所示，本文所需的感知内容主要分为如下四个主要部分和三个子部分：

主要部分：

- 竞技场 (Arena): 包括防御塔及双方所部署的部队和建筑物。
- 手牌：位于竞技场下方，包括当前卡片图像及其所需的圣水 (Elixir)。
- 游戏时间：当前阶段的剩余时间。
- 总圣水：用于卡牌的使用，随时间自动恢复。

子部分：

- 主塔：处于三个防御塔中间，当其被摧毁时，对局直接结束。
- 副塔：处于左右两侧的防御塔。
- 部队：玩家通过卡牌部署的战斗单位，包括可移动地面和空中单位，以及建筑单位。



图 1-1 右上角为当前阶段的剩余时间；中间部分为竞技场，其上有双方所部署的部队及建筑；下方显示了可用手牌以及部署所需的圣水，最下方显示了当前的可用圣水总量。

<sup>①</sup> <https://github.com/wty-yy/katacv>, 复现内容包括：LeNet-5、AlexNet、VGG16、GoogleNet、ResNet50、YOLOv1、YOLOv3、YOLOv4、YOLOv5、CTCLoss & CRNN、miniGPT

**游戏目标** 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 (Regular Time): 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 (Over Time): 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。

**游戏过程** 一次对局中每个玩家牌库大小为固定的 8 张，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，实时决策使用卡牌的类别和位置，采取进攻或防守策略，最终按照上述“游戏目标”给出的规则判断胜负。

## 1.4 本毕设的贡献

在本文中，首次提出一种基于非嵌入式的离线强化学习设计方案，应用于游戏《皇室战争》(Clash Royale, CR)，并最终在移动端设备上进行实时决策验证其可行性。设计方案如下：

1. 生成式数据集：本文设计了一个与本任务相关的生成式数据集，使其可以模仿真实场景生成含有任意数量部队及种类的带有识别标签的图像，能够非常高效地生成用于目标识别的训练数据集。
2. 感知融合：本文利用 YOLOv8<sup>[29]</sup> 并结合目标追踪技术 ByteTrack<sup>[30]</sup> 实现了对视频流中部队位置及类别的实时识别，通过 PaddleOCR 完成了对数字及时间信息的识别，利用 ResNet 完成对卡牌和圣水图标的分类，结合上述几种感知模型，本文还提出了一系列针对于本任务的上下文信息过滤方法用于优化感知特征。
3. 模型决策：本文使用离线强化学习算法 StARformer 作为决策模型，设计相应的特征输入结构以及损失函数，并将难以学习的离散动作序列转化为连续动作序列进行训练，此方法提高了模型的决策频率与决策准确率，最后部署在手机上完成实时对局决策。

离线强化学习模型使用手工录制的 11 万帧专家数据集进行训练，能够战胜游戏内置 AI，虽然无法达到 100% 胜率，但本研究发现该离线强化学习模型能够展现出一定的模仿及泛化能力，从一定程度上说明了这种非嵌入式离线强化学习算法的可行性。

## 2 目标识别数据集搭建及模型架构设计

本毕设的首要任务是完成图 1-1 中竞技场部分的目标识别问题，因为决策所需的状态信息主要来自于该部分，对于竞技场中第  $i$  各单位  $u_i$ ，其所包含的信息可以表示为  $u = (\mathbf{x}, \text{cls}, \text{bel}, \text{bar}_1, \text{bar}_2)$ ，其中  $\mathbf{x}$  表示  $u$  所在的网格坐标， $\text{cls}$  表示  $u$  所属的部队类别， $\text{bel}$  表示  $u$  所属的派别， $\text{bar}_1, \text{bar}_2$  表示  $u$  当前的生命值及其余条状图像信息。

在本章节将介绍如何使用目标识别模型完成对每个单位中  $\mathbf{x}, \text{cls}, \text{bel}$  信息的识别，由于训练目标识别模型需要基于大量的有标记图像，而该任务没有任何相关的开源数据集，若逐帧人工标记效率极低且成本高昂，因此本毕设基于该任务提出一种高效的带标记图像的生成方案，并目标识别代码使其可以有效训练该生成式数据集以满足上述识别要求，识别数据集制作以及模型更新流程如图 2-1 所示。



图 2-1 目标识别数据集制作流程：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，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。

### 2.1 切片数据集制作流程

切片数据集中制作本毕设使用了一个强大的通用计算机视觉模型：Segment Anything Model (SAM)<sup>[31]</sup>，该模型为 Meta AI 公司于 2023 年开源，该模型可以根据输入的提示信息生成对象的分割掩码，也可对图像中的所有对象生成掩码，其是在一个包

含 1100 万张图像和 11 亿各掩码的数据集上进行的训练，能够在很多零样本任务上体现出强大的性能，SAM 模型的核型框架包含如下三个部分：

1. 视觉编码器（Vision Encoder）：基于 ViT 的图像特征提取架构，用于捕获图像的上下文信息，将输入图像转化为特征编码。
2. 提示编码器（Prompt Encoder）：用于处理输入中的提示信息，提示信息分为离散（点、框选区域、文本信息）和连续（掩码）两种，分别使用位置嵌入（Positional Encoding）和 CNN 对提示信息进行编码。
3. 解码器（Decoder）：将视觉编码器和提示篇马奇生成的特征进行融合，并输出提示信息所对应的分割掩码。

由于生成式数据集需要获取每一个单位不同角度的切片图像，人工创建分割掩码过于繁琐，但单位的目标框容易标注，所以考虑使用 SAM 中框选区域的提示方式生成对象掩码，在图 2-1 的中上以及右上部分展示了使用 SAM 制作切片数据集的过程。

## 2.2 目标识别模型设计

由于需要追求高效的识别速度，所以本文使用了一阶段识别器 YOLOv8<sup>[29]</sup>，下面对 YOLO 系列模型架构进行分析，并给出本毕设对其进行修改的部分。

**定义 2.1 (边界框， Boudning Box):** 设正实数  $x, y, w, h$ ，称四元组  $B = (x, y, w, h)$  为边界框，其中  $(x, y)$  表示当前边界框中心， $(w, h)$  表示边界框的宽度和高度。 ◇

**定义 2.2 (交并比， Intersection over Union, IOU):** 对于任意两个边界框  $B_1, B_2$ ，分别用  $S_1, S_2$  表示其所围住的点集，则称  $B_1, B_2$  的交并比为  $\text{IOU}_{B_2}^{B_1} := \frac{S_1 \cap S_2}{S_1 \cup S_2}$  ◇

设  $S, B, C \in \mathbb{Z}^+$  分别表示图像网格化大小、每个网格中边界框预测框数目、总分类类别数，通过 CNN 可以将图像空间进行降维，从而得到对应的不同网格化大小。例如，在如图 2-2 中原图像宽高为  $416 \times 416$ ，CNN 中每次降采样会将图像的宽高均减小一半，于是经过步长（Stride）为  $s = 2^3$  的降采样就可以得到左图中  $(H/s) \times (W/s) =: S \times S$  即  $52 \times 52$  的网格大小，每个网格中的感受野大小（即步长）即为对应网格中  $2^3 \times 2^3$  个像素，目标识别模型需要在每个网格处做出  $B$  个边界框框预测。

图 2-2 中，对于每个目标框  $B$ ，其中心点用黄色点标出，假设该中心点位于网格大小为  $52 \times 52$  中的第  $(i, j)$  网格内，则在模型预测的输出中，也应该由  $(i, j)$  网格对应的边界框进行预测。在 YOLOv3<sup>[32]</sup> 中，将 Fast-RCNN<sup>[24]</sup> 中锚框的概念引入 YOLO 系列，对于不同的网格划分，对应分配不同的尺度估计框，称为锚框（Anchor Bounding Box），这些锚框是基于数据集中的全体锚框做 K 近邻得到（距离衡量标准为负的相对交并比），可以将数据集中识别框大小的先验信息引入到模型中。按照网格划分的从大到小，分别分配  $B$  个从小到大的锚框，因为更大的网格对应更高的分辨率，其具有更多的细节信息，从而可以识别小目标框，所以给其分配更小的锚框，反之亦然。

**定义 2.3 (相对预测框):** 假设当前网格步长为  $s$ ，对应的一个锚框为  $(A_w, A_h)$ ，则

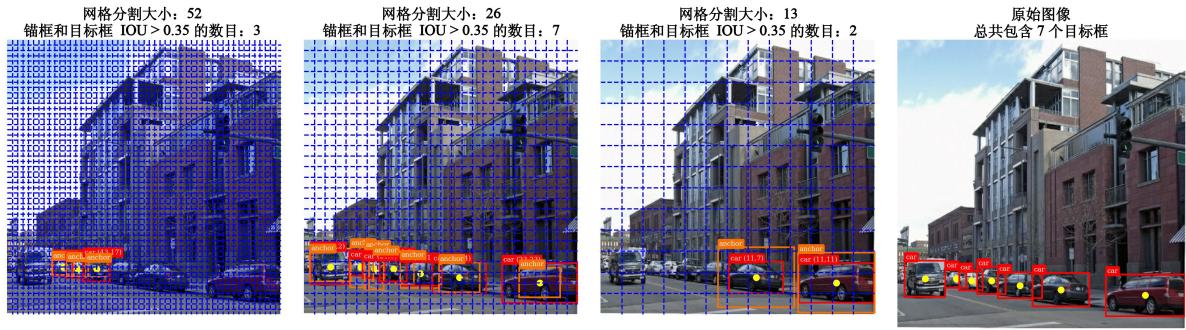


图 2-2 3 个不同网格大小下的目标框与锚框，红色为目标框，橙色为基于数据集分配的锚框，在 3 个不同的网格大小下各分配了 3 种锚框，在图中只显示了相对  $\text{IOU} > 0.35$  的锚框。演示的标记图片来自于 PASCAL VOC 数据集<sup>[33]</sup>。

网格  $(i, j)$  处的相对预测框定义为六元组  $(x, y, w, h, c, \{p_c\}_{c=1}^C, \text{bel})$ ，其中：

- $(x, y)$  表示当前预测框中心点相对于步长  $s$  的比例，偏移量为  $(i \cdot s, j \cdot s)$ 。
- $(w, h)$  表示当前预测框的长宽相对于锚框  $(A_w, A_h)$  的比例。
- $c$  表示当前预测的置信度，定义为  $\Pr(\text{Object}) \cdot \text{IOU}_{\text{pred}}^{\text{true}}$ 。
- $p_c$  表示当前预测框预测为类别  $c$  的概率。
- $\text{bel}$  表示当前预测框预测从属派别为敌方的概率。 ◇

从上述定义中不难看出，当前网格步长  $s$  下网格  $(i, j)$  处的相对预测框所对应的全局边界框为  $((j + x)s, (i + y)s, wA_w, hA_h)$ 。传统目标识别通常为单类别识别，本毕设任务由于包含单位的从属派别类别，故为多类别识别问题，需要对预测结果以及对应损失函数进行修改。

**定义 2.4 (交叉熵损失, Cross-Entropy Loss, CE Loss):** 设模型的输出为  $\hat{z} \in \mathbb{R}^N$ ，则模型预测分布为（Softmax 变换）

$$\hat{y}_i = \text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}} \quad (2-1)$$

对于  $N$  维离散目标分布为  $y \in \mathbb{R}^N$ ，多元交叉熵损失为

$$\mathcal{L}_{CE}(\hat{y}, y) = - \sum_{i=1}^N y_i \log \hat{y}_i \quad (2-2)$$

特殊地，当  $N = 2$  时，不妨令目标分布为  $\{y, 1 - y\}$ ，则式 2-2 被称为二元交叉熵损失 (Binary Cross-Entropy Loss, BCE Loss)

$$\mathcal{L}_{BCE}(\hat{y}, y) = y \cdot \log \hat{y}_1 + (1 - y) \cdot \log \hat{y}_2 \quad (2-3)$$

◇

在目标识别任务中目标分布  $y$  通常为独热分布 (Onehot)，其中正确预测的概率为 1，其余类别概率为 0，可以表示  $\text{onehot}(c) = \{[i = c]\}_{i=1}^C$ ，[条件] 表示当内部条件成立时为 1，否则为 0。然而上述 Softmax 函数只有当  $z_i \gg z_j (i \neq j)$  时接近该分布，这会导致模型

对其预测过于自行，在目标识别任务中容易出现过拟合现象<sup>[34]</sup>。

**定义 2.5(目标识别损失函数):** 设当前网格步长为  $s$ , 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  $(\widehat{\text{box}}_{ijk}, \hat{c}_{ijk}, \hat{p}_{c_{ijk}}, \widehat{\text{bel}}_{ijk})$ , 对应的预测框  $(\text{box}_{ijk}, \text{cls}_{ijk}, \text{bel}_{ijk})$ , 下面给出该步长下的损失函数

$$\begin{aligned} \mathcal{L}_s(\hat{y}, y) = & \sum_{i=1}^{H/s} \sum_{j=1}^{W/s} \sum_{k=1}^B \mathbb{1}_{ijk}^{noobj} \lambda_{noobj} \mathcal{L}_{\text{BCE}}(\hat{c}_{ijk}, 0) \\ & + \mathbb{1}_{ijk}^{obj} \left[ \lambda_{\text{box}} \mathcal{L}_{\text{CIOU}}(\widehat{\text{box}}_{ijk}, \text{box}_{ijk}) + \lambda_{obj} \mathcal{L}_{\text{BCE}}(\hat{c}_{ijk}, \text{IOU}_{pred}^{true}) \right. \\ & + \lambda_{\text{class}} \sum_{c=1}^C \mathcal{L}_{\text{BCE}}\left(\{\hat{p}_{ijk}\}_c, \text{onehot}(\text{cls}_{ijk})_c\right) \\ & \left. + \lambda_{\text{class}} \mathcal{L}_{\text{BCE}}(\widehat{\text{bel}}_{ijk}, \text{bel}_{ijk}) \right] \end{aligned} \quad (2-4)$$

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 Complete-IOU 损失<sup>[35]</sup> 是基于 IOU 损失的一种改进, 能够加速边界框回归预测问题中对 MSE 损失的改进。 ◇

由于本任务的类别数目超过 150 种, 而当前通用目标识别数据集 COCO<sup>[36]</sup> 类别为 80 种, 所以本文使用了双模型识别器, 如图 2-1 中左下角部分, 多个检测器进行组合的模型预测效果见第五章中表 5-1。

## 2.3 生成式目标识别数据集

假设将每个切片作为绘制单位, 定义  $u = (\text{img}, \text{box}, \text{level})$ , 其中  $\text{img}$  为绘制单位的切片图像;  $\text{box} = (x, y, w, h, \text{cls}, \text{bel})$  为绘制单位的边界框参数,  $(x, y)$  为切片中心点位于图像的二维坐标,  $(w, h)$  为切片的宽高大小,  $\text{cls}$  为当前切片的所属类别,  $\text{bel}$  为当前切片的所属派别;  $\text{level}$  为当前切片的所属图层等级, 图层等价划分如表 2-1 所示。

表 2-1 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 2-1 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

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 A.1）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  $\alpha$ ，待识别类别集合  $C$

**输出：**image, box

```

1 image ← 空图像, box ← {} // 初始化参数
2  $U \leftarrow \{u_i \in U : u_i^{level} > u_j^{level}, \forall i, j \in \{1, \dots, |U|\} \text{ 且 } i < j\}$ 
3 while True do
4     mask ← 空掩码,  $U_{avail} \leftarrow U$ 
5     for  $i = 1, 2, \dots, |U|$  do
6         if  $\frac{u_i^{img} \cap mask}{u_i^{img}} > \alpha$  then
7              $| U_{avail} \leftarrow U_{avail} - R(u_i)$  // 删除与  $u_i$  相关的单位
8         end
9         mask ← mask  $\cup u_i^{img}$ 
10    end
11    if  $|U_{avail}| = |U|$  then
12        break // 覆盖单位筛选完成
13    end
14     $U \leftarrow U_{avail}$ 
15 end
16  $U \leftarrow \{u_i \in U : u_i^{level} < u_j^{level}, \forall i, j \in \{1, \dots, |U|\} \text{ 且 } i < j\}$ 
17 for  $i = 1, 2, \dots, |U|$  do
18     img ← img  $\cup u_i^{img}$  // 图像绘制
19     if  $u_i^{cls} \in C$  then
20         box ← box  $\cup u_i^{box}$  // 边界框保存
21     end
22 end

```

---

算法 2-1 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  $\alpha$ ，当被覆盖单位面积超过该单位切片面积的  $\alpha$  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高到低进行绘制，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 2-1。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  $\alpha$ ，可以得到如图 2-3 所示的生成结果<sup>①</sup>。



图 2-3 生成式数据集实例

<sup>①</sup> 生成式代码见附录B.1

## 3 感知融合

### 3.1 感知模型

#### 3.1.1 目标识别模型

本文使用 YOLO 系列模型作为本次目标识别模型，分别尝试了 YOLOv5<sup>[37]</sup> 和 YOLOv8<sup>[29]</sup> 作为目标识别器，下面分别对其模型架构改进进行介绍：

YOLOv5 主要是对 YOLOv4<sup>[38]</sup> 模型进行了少许改进，YOLOv4 模型在 Backbone 部分（初步特征提取）的主要改进是使用了基于跨阶段部分网络（Cross Stage Partial Networks, CSPNet）<sup>[39]</sup> 结构的 DarkNet<sup>[32]</sup> 并在输出特征时使用空间金字塔池化（Spatial Pyramid Pooling, SPP）<sup>[40]</sup> 对特征进行不同尺度上的提取，在 Neck 部分（进一步特征提取）中使用了路径聚合网络（Path Aggregation Network, PANet）<sup>[41]</sup>，结合特征金字塔与路径压缩，缩短了较低层与最顶层特征之间的信息路径，能够有效进行特征融合。

而 YOLOv8 模型结构上仍然使用 CSP 和 SPP 进行特征提取，相对 YOLOv5 使用了更多的工程优化方法对模型预测速度进行大幅优化，Head 部分（预测框输出）则重新回到 YOLOv1<sup>[21]</sup> 无锚框预测的方法，使预测框不再受到锚框约束。

ByteTrack<sup>[30]</sup> 是一种基于 Kalman 滤波的多目标跟踪（Multiple Object Tracking, MOT）算法，首先需要分别对不同边界框作为跟踪对象，建立独立的线性运动方程，通过 Kalman 滤波结合之前帧的追踪信息，以递归的方式给出目标边界框在当前时刻下的位置预测，再通过计算交并比（定义 2.1）定位当前帧下跟踪对象的位置。与传统 MOT 算法不同的是，ByteTrack 尝试利用滤波预测的方法，从更低的置信度对应的边界框中找到可能被忽略的跟踪边界框，从而恢复可能因遮挡等原因被低估的边界框。

#### 3.1.2 光学字符识别模型

PaddleOCR<sup>[28]</sup> 是百度公司研发并开源的一个基于 PaddlePaddle 框架的光学字符识别（Optical Character Recognition, OCR）系统，PaddleOCR 提供了一整套从图像预处理到文字识别的方案，其模型架构主要分为如下两个模块：

1. 文本检测模块：使用 ResNet<sup>[15]</sup>、MobileNet<sup>[42]</sup> 等经典的卷积神经网络作为特征提取网络，使用特征金字塔网络（Feature Pyramid Networks, FPN）<sup>[43]</sup> 等方法来融合多尺度特征，使用可微分二值化（Differentiable Binarization, DB）<sup>[44]</sup> 等方法来预测文本框。
2. 文本识别模块：使用卷积循环神经网络（Convolutional Recurrent Neural Networks, CRNN）<sup>[25]</sup> 提取文字区域的特征，首先使用卷积网络对图像进行特征提取，再使用 Bi-LSTM<sup>[27]</sup> 等序列模型来捕捉字符之间的依赖关系，最后使用 CTC（Connectionist Temporal Classification）<sup>[26]</sup> 作为文字序列的损失函数。

### 3.1.3 图像分类模型

本毕设的图像分类模型使用 ResNet 结构，其核心思想是通过引入短连接（Shortcut Connections），使得网络可以直接学习残差（Residual），从而更容易学习到恒等映射，缓解深度神经网络中梯度消失的问题。具体来说，ResNet 使用残差块（Residual Block）来实现上述操作，残差块两部分构成，卷积层：卷积变换  $F_W(\cdot)$ 、批归一化（Batch Normalization, BN）<sup>[45]</sup> 以及 ReLU 激活函数；残差连接：将输入直接加在卷积层的输出结果上。残差块结构可以表示为

$$y = \sigma(\text{BN}(F_W(x))) + x \quad (3-1)$$

其中  $x$  为输入图像特征， $y$  为输出图像特征， $F_W$  是以卷积核  $W$  的卷积变换，BN 为批归一化操作， $\sigma(x) = \frac{x+|x|}{2}$  为 ReLU 函数。

## 3.2 感知特征提取

在感知特征提取中，本文将同时用到上述三种模型作为一阶段图像特征提取，分别可以得到当前时刻下三个预处理信息：剩余时间（OCR）、竞技场中的预测框（YOLO）、当前手牌及总圣水（图像分类器）。下面将介绍特征提取器的设计方法，可以将预处理信息进一步转化为决策模型的输入信息，它们分别为环境状态信息提取（State）、执行动作信息提取（Action）和环境奖励信息提取（Reward）。

### 3.2.1 状态特征提取

状态特征包括四种信息：

- 经过的总时长：直接通过 OCR 对图 1-1 右上角的阶段剩余时间信息识别后，简单处理即可。
- 竞技场中部队信息：每个部队由五个参数构成二维位置坐标、类别、派别、生命值、其余条状信息构成  $u := (x, \text{cls}, \text{bel}, \text{bar}_1, \text{bar}_2)$ 。
- 手牌信息：由于手牌可以被拖出但不释放，因此仅识别手牌图像无法确定其真实状态，需要通过是否做出动作来判断当前手牌是否被使用，故要基于动作特征 3.2.2 来判断当前真实手牌信息。
- 圣水信息：通过 OCR 对图 1-1 下方可用圣水中的数字进行识别。



图 3-1 当模型在第 1 帧关联了部队 1 与等级、生命值信息的对应关系，通过目标追踪及上下文信息记忆，即使在第 2,3 帧未能检测到部队 1，通过等级或生命值的关联性推理，模型同样可以推理得到当前单位的真实位置。

在竞技场中部队信息特征提取时，本文引入一种基于上下文关联性推理的方式来解决部队识别错误或漏识别的问题，在本任务中由于等级和生命值信息容易识别，每个部队都存在一个与之唯一对应的等级和生命值信息，且部队与等级或生命值信息的相对位置基本不变，所以当部队识别框消失、类别识别错误、派别识别错误时，利用之前与之关联的等级和生命值信息中进行修正，实现效果如图 3-1 所示。

需要注意状态处理的细节问题，在执行动作之前，状态特征中应不包含任何直接与该动作相关的先验信息，否则模仿学习可以通过该先验信息直接给出动作预测，而真实环境中由于不再出现这类动作的先验信息，所以导致模型无法做出决策，错误状态信息如右图 3-2 所示。

假设目标识别每个预测框的识别结果包括七个参数  $b := (x, y, w, h, \text{id}, \text{cls}, \text{bel})$ ，分别为预测框的中心点  $(x, y)$ ，宽高  $(w, h)$ ，目标追踪编号  $\text{id}$ ，预测所属类别  $\text{cls}$  以及所属派别  $\text{bel}$ 。

**定义 3.1 (计数记忆缓存):** 设  $I \subset \mathbb{N}$  为目標追踪指标集， $S$  为可数记忆元素集合， $V \subset \mathbb{R}^+$  为可用时间集合，定义  $M_T : I \times S \rightarrow \mathbb{Z}^+ \times V$  为缓存大小为  $T$  的计数记忆缓存，其满足  $\forall \text{id} \in I, s \in S, (n, t) := M_T(\text{id}, s)$ ，在时间间隔  $[t - T, t] \cap V$  下追踪指标为  $\text{id}$  的预测框中，存在  $n$  个预测框包含元素  $s$ 。 ◇

初始化派别记忆缓存  $M_T^{\text{bel}}$  和类别记忆缓存  $M_T^{\text{cls}}$ ，其记忆元素集合分别为派别集合  $\{0, 1\}$  及全体类别集合，假设当前时间为  $t$ ，于是状态特征提取可以分为下述 6 步<sup>①</sup>：

1. 更新派别记忆缓存：对当前每个预测框  $b$ ，更新  $M_T^{\text{bel}}(b^{\text{id}}, b^{\text{bel}})$ ，并对  $b^{\text{bel}}$  进行修正，使得  $M_T^{\text{bel}}(b^{\text{id}}, b^{\text{bel}})_1 = \max \{M_T^{\text{bel}}(b^{\text{id}}, x)_1 : x \in \{0, 1\}\}$ 。
2. 全局文本信息查找：使用 OCR 对当前竞技场中全部文本信息进行识别，用于解决图 3-2 中未放置单位的错误识别问题。
3. 等级、生命值及防御塔信息关联：将等级与部队生命值、防御塔与防御塔生命值进行关联。
4. 部队信息关联：基于贪心的关联策略，从下至上，将部队与在一定范围内最近的等级和生命值进行关联，并基于步骤 2 结果，去除具有对应文本的部队单位。
5. 缓存清理：将  $M_T^{\text{bel}}, M_T^{\text{cls}}$  中超出时间间隔  $[t - T, t]$  的信息置零。
6. 更新类别记忆缓存：结合关联性信息，对部队及其关联的等级和生命值信息所对应的  $M_T^{\text{cls}}$  进行更新，同时对部队类别进行修正，更新及修正方法与步骤 1 类似。

<sup>①</sup> 状态特征提取部分代码见附录 B.2.1。



图 3-2 动作直接导致的错误状态先验信息：第 1 帧中目标识别错误识别到未部署的部队状态；第 2 帧中 OCR 识别器在目标识别识别到动作之前，产生了错误的总圣水识别信息，因此需要将动作帧提前到该帧之前；第 3 帧中目标识别成功识别到圣水动画，可以判断玩家在该时刻之前部署了部队。

假设目标识别每个预测框的识别结果包括七个参数  $b := (x, y, w, h, \text{id}, \text{cls}, \text{bel})$ ，分别为预测框的中心点  $(x, y)$ ，宽高  $(w, h)$ ，目标追踪编号  $\text{id}$ ，预测所属类别  $\text{cls}$  以及所属派别  $\text{bel}$ 。

**定义 3.1 (计数记忆缓存):** 设  $I \subset \mathbb{N}$  为目標追踪指标集， $S$  为可数记忆元素集合， $V \subset \mathbb{R}^+$  为可用时间集合，定义  $M_T : I \times S \rightarrow \mathbb{Z}^+ \times V$  为缓存大小为  $T$  的计数记忆缓存，其满足  $\forall \text{id} \in I, s \in S, (n, t) := M_T(\text{id}, s)$ ，在时间间隔  $[t - T, t] \cap V$  下追踪指标为  $\text{id}$  的预测框中，存在  $n$  个预测框包含元素  $s$ 。 ◇

初始化派别记忆缓存  $M_T^{\text{bel}}$  和类别记忆缓存  $M_T^{\text{cls}}$ ，其记忆元素集合分别为派别集合  $\{0, 1\}$  及全体类别集合，假设当前时间为  $t$ ，于是状态特征提取可以分为下述 6 步<sup>①</sup>：

1. 更新派别记忆缓存：对当前每个预测框  $b$ ，更新  $M_T^{\text{bel}}(b^{\text{id}}, b^{\text{bel}})$ ，并对  $b^{\text{bel}}$  进行修正，使得  $M_T^{\text{bel}}(b^{\text{id}}, b^{\text{bel}})_1 = \max \{M_T^{\text{bel}}(b^{\text{id}}, x)_1 : x \in \{0, 1\}\}$ 。
2. 全局文本信息查找：使用 OCR 对当前竞技场中全部文本信息进行识别，用于解决图 3-2 中未放置单位的错误识别问题。
3. 等级、生命值及防御塔信息关联：将等级与部队生命值、防御塔与防御塔生命值进行关联。
4. 部队信息关联：基于贪心的关联策略，从下至上，将部队与在一定范围内最近的等级和生命值进行关联，并基于步骤 2 结果，去除具有对应文本的部队单位。
5. 缓存清理：将  $M_T^{\text{bel}}, M_T^{\text{cls}}$  中超出时间间隔  $[t - T, t]$  的信息置零。
6. 更新类别记忆缓存：结合关联性信息，对部队及其关联的等级和生命值信息所对应的  $M_T^{\text{cls}}$  进行更新，同时对部队类别进行修正，更新及修正方法与步骤 1 类似。

<sup>①</sup> 状态特征提取部分代码见附录 B.2.1。

### 3.2.2 动作特征提取

动作特征包含两种信息：

- 当前执行动作的二维坐标  $x$ 。
- 当前执行动作所用的卡牌编号  $\text{card} \in \{1, 2, 3, 4\}$ 。

在进行动作特征提取时，需通过目标识别模型识别到的圣水预测框来判断执行动作的时刻以及坐标位置，对圣水预测框的目标检测如图 3-2 中第 3 帧所示，但是在第 2 帧中已经出现了圣水对象，动作执行本应该发生在第 2 帧，但是模型无法对其进行识别导致动作判断出现延迟，所以需通过第 2 帧中总圣水识别发生突变来进行判断动作的执行，因此需要将第 3 帧识别到的动作前移至第 2 帧上。

还需结合当前手牌及圣水上方向的文字信息来判断使用的卡牌编号，通过维护一个手牌记忆缓存记录当前可用手牌，当手牌被玩家或智能体拖出牌库，则将其加入到候选手牌中，每次动作执行的卡牌编号将文本与候选手牌进行比对得到，当两个文本串的 Levenshtein 编辑距离<sup>[46]</sup>不超过 2 时，则认为两个文本串相同。

初始化圣水记忆缓存  $M_T^{\text{elixir}}$ ，其记忆元素集合为可部署单位的二维坐标空间，假设当前时间为  $t$ ，于是动作特征提取可以分为下述 4 步<sup>②</sup>：

1. 缓存清理：将  $M_T^{\text{elixir}}$  中超出时间间隔  $[t - T, t]$  的信息置零。
2. 更新可用手牌及候选手牌：通过手牌分类器的识别结果以及缓存中记录的手牌，可以完成候选手牌、可用手牌的信息更新。
3. 记录总圣水的突变时刻：通过 OCR 识别当前总圣水以及上一帧总圣水，可用判断当前的总圣水是否发生减少或者无法识别的情况，当此类情况发生则认为动作执行可能发生在当前帧。
4. 动作查找：通过竞技场中对圣水单位的目标识别，判断动作执行的位置，动作执行的真实时刻为一段时间内最早发生的总圣水突变时刻，动作卡牌编号需要结合 OCR 文本识别与候选手牌集合判断，最后再对当前可用手牌进行更新。

### 3.2.3 奖励特征提取

奖励特征仅包含奖励  $r \in \mathbb{R}$  一种信息，通过 OCR 识别可以得到敌我防御塔具体生命值，敌我主、副塔如图 1-1 中所示，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， $\Delta h_i^{\text{bel}}$  表示前一帧与当前帧生命值的差值， $H_i^{\text{bel}}$  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

$$r_{tower} = \sum_{\text{bel}=0}^1 \sum_{i=0}^2 (-1)^{\text{bel}+1} \frac{\Delta h_i^{\text{bel}}}{H_i^{\text{bel}}} \quad (3-2)$$

<sup>②</sup> 动作特征提取部分代码见附录 B.2.2。

- 
- 2. 防御塔摧毁奖励  $r_{destory}$ : 当敌我副塔被摧毁时给予  $(-1)^{bel+1}$  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  $(-1)^{bel} 0.1$  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:

$$r = r_{tower} + r_{destory} + r_{activate} + r_{elixir} \quad (3-3)$$

需要注意的是, 当图像中的数字信息出现噪声干扰或产生错误目标位置识别时, 需要终止奖励的错误更新, 例如: 部署单位时其他部件对其产生的遮挡 (等级、生命值、圣水、时钟、文本) <sup>③</sup>。

---

<sup>③</sup> 奖励特征提取部分代码见附录 B.2.3。

## 4 决策模型

### 4.1 离线强化学习

离线强化学习（Offline Reinforcement Learning）是指在没有与环境交互的情况下，使用预先手机的随机或专家数据来训练强化学习算法，这类算法适用于无法频繁与环境交互或交互代价高昂的场景，例如自动驾驶、机器人控制等。由于本任务中的非嵌入环境，与环境交互速度效率很低，所以考虑使用离线强化学习作为决策模型。

首先对强化学习中的概念进行介绍，考虑无限长度的 Markov 决策过程（MDP），定义为  $(\mathcal{S}, \mathcal{A}, p, r, \rho_0)$ ，其中  $\mathcal{S}$  为状态空间， $\mathcal{A}$  为动作空间， $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$  为状态转移方程， $r : \mathcal{S} \rightarrow \mathbb{R}$  为奖励函数， $\rho_0 : \mathcal{S} \rightarrow \mathbb{R}$  为初始状态  $s_0$  对应的分布。

令  $\pi$  表示决策函数  $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ ，令  $R(\pi)$  表示其期望所获得的总奖励（回报）：

$$R(\pi) = \mathbb{E}_{S_1, A_1, S_2, A_2, \dots} \left[ \sum_{t=0}^{\infty} r(S_t) \right], \quad \text{其中 } S_1 \sim \rho_0(\cdot), A_t \sim \pi(\cdot | S_t), S_{t+1} \sim p(\cdot | S_t, A_t) \quad (4-1)$$

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，而下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。

#### 4.1.1 Decision Transformer

Decision Transformer (DT)<sup>[8]</sup> 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)

$$\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T, a_T, r_T, s_{T+1}) \quad (4-2)$$

其中  $s_{T+1}$  为终止状态，则  $\rho$  可以视为建模为序列

$$R_0, s_1, a_1, R_1, s_2, \dots, a_{T-1}, R_{T-1}, s_T, a_T \quad (4-3)$$

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

DT 模型中序列编码模型使用的是 GPT 模型<sup>[47]</sup>，即仅含有编码器的因果注意力机制。具体来讲，假设当前处理的序列  $X \in \mathbb{R}^{d \times N}$  长度为  $N$ ，每个特征编码维度为  $d$ ，则注意力机制<sup>[48]</sup> 包含三个可学习矩阵  $Q_\theta, K_\theta \in \mathbb{R}^{d_k \times d}, V_\theta \in \mathbb{R}^{d_v \times d}$ ，分别对应生成询问键 (Query)，查询键 (Key) 和价值键 (Value)

$$Q = Q_\theta X, \quad K = K_\theta X, \quad V = V_\theta X, \quad (4-4)$$

则对于序列中第  $i \in \{1, \dots, N\}$  个特征对应的交叉注意力 (Cross-Attention) 为

$$\mathbf{z}_i^{cross} = \sum_{j=1}^N \text{softmax} \left( \left\{ \frac{1}{\sqrt{d_k}} \langle \mathbf{q}_i, \mathbf{k}_l \rangle \right\}_{l=1}^N \right)_j \cdot \mathbf{v}_j \iff Z^{cross} = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (4-5)$$

其中包含系数  $1/\sqrt{d_k}$  的原因：不妨假设  $\mathbf{q}_{ij}, \mathbf{k}_{lj} \sim \mathcal{N}(0, \sigma^2)$ , ( $j \in \{1, \dots, d_k\}$ ), 则  $\sum_{j=1}^{d_k} \mathbf{q}_{ij} \mathbf{k}_{lj} \sim \mathcal{N}(0, d_k \sigma^2)$ , 由于初始化的神经网络输出可以保证  $\sigma \approx 1$ , 因此使用系数  $1/\sqrt{d_k}$  可以保持输出的方差在 1 左右, 避免发散。

**因果注意力 (Causal-Attention)** 为 (每个特征  $i$  只能看到  $j \leq i$  的特征)

$$\mathbf{z}_i^{causal} = \sum_{j=1}^i \text{softmax} \left( \left\{ \frac{1}{\sqrt{d_k}} \langle \mathbf{q}_i, \mathbf{k}_l \rangle \right\}_{l=1}^N \right)_j \cdot \mathbf{v}_j \quad (4-6)$$

$$\iff Z^{causal} = \text{softmax} \left( \frac{(QK^T) \odot M}{\sqrt{d_k}} \right) V, \quad \text{其中 } M \text{ 为 } N \text{ 阶下三角阵} \quad (4-7)$$

引入因果注意力机制后, 每个特征由于无法观察到后续特征, 所以可以对相邻的下一个特征进行预测, 从而无需再对编码器和解码器进行区分, 降低了代码复杂性。

**DT 训练方法:** 首先从离线数据集中随机采样得到一段长度为  $N$  的轨迹

$$\tau_{t-N+1:t} =: (s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_N, a_N, r_N) = \{(s_n, a_n, R_n)\}_{n=1}^N \quad (4-8)$$

再每个特征编码到相同维度  $\mathbb{R}^d$  下, 按式 4-3 建模为长度  $3N$  的序列  $\tau \in \mathbb{R}^{3N \times d}$ , 用 GPT 模型对序列进行特征编码可以得到和  $\tau$  维度相同的编码序列  $\tau'$ , 再取出状态序列所对应的编码结果, 通过线性变换映射到动作空间维度, 从而得到对相邻动作的预测

$$\tau'_2, \tau'_5, \dots, \tau'_{3k-1}, \dots, \tau'_{3N-1} \xrightarrow{\text{线性变换}} \hat{a}_1, \hat{a}_2, \dots, \hat{a}_k, \dots, \hat{a}_N \quad (4-9)$$

对于离散动作空间使用定义 2.4 中的多元交叉熵损失, 连续动作空间则使用  $\ell^2$  范数 (Mean Square Error, MSE) 作为损失函数。

**DT 验证方法:** 需给出模型期望达到的总奖励  $\hat{R}_0$ , 通过自迭代的方式时模型完成动作预测, 具体来说, 假设初始状态为  $s_1$ , 则初始轨迹为  $\tau_1 = (\hat{R}_0, s_1)$ , 模型对  $\tau_1$  进行编码, 取出最后一个状态  $s_1$  所对应的预测动作  $\hat{a}_1$  与环境交互, 得到新的状态  $s_2$  和奖励  $r_1$ , 令  $\hat{R}_1 = \hat{R}_0 - r_1$ , 从而得到新的序列  $\tau_2 = (\hat{R}_1, s_1, \hat{a}_1, \hat{R}_1, s_2)$ , 模型再对  $\tau_2$  进行编码, 取出  $s_2$  所对应的预测动作  $\hat{a}_2$  与环境交互, 以此类推, 直到环境达到终止状态为止。这种方式和自然语言模型中文本生成的做法基本一致。

### 4.1.2 StARformer

StARformer<sup>[9]</sup> 是一种基于 ViT<sup>[17]</sup> 专门对状态中的图像特征编码进行的改进, 通过将轨迹  $\tau$  中的  $\{(a_{n-1}, r_{n-1}, s_n)\}_{n=1}^{3N}$  ( $a_0$  使用特征填充补全) 按照空间维度进行展开 ( $s_n$  使用图像分块的方法将图像分块序列化), 并压缩到空间特征维度  $d'$ , 进而在**空间维度**上使用交叉注意力机制进行编码, 得到  $\{(a'_{n-1}, r'_{n-1}, s'_n)\}_{n=1}^{3N}$ , 再将第  $n$  时刻对应的编码  $(a'_{n-1}, r'_{n-1}, s'_n)$  全部展平, 使用线性变换到时间维度  $\mathbb{R}^d$  中, 将其记为  $\{l_n\}_{n=1}^N$ , 则时

间维度的序列建模为

$$\tau := s_1, l_1, s_2, l_2, \dots, s_N, l_N \quad (4-10)$$

通过因果注意力机制（相邻的  $(s_n, l_n)$ ,  $n = 1, \dots, N$  之间仍然具有注意力机制）可以完成对时序序列信息的编码，将编码结果记为  $\tau'$ ，类似 DT 的预测方法，只考虑所有状态对应的编码结果，通过线性变换映射到动作空间维度，从而得到相邻动作的预测

$$\tau'_1, \tau'_3, \dots, \tau'_{2k-1}, \dots, \tau'_{2N-1} \xrightarrow{\text{线性变换}} \hat{a}_1, \hat{a}_2, \dots, \hat{a}_k, \dots, \hat{a}_N \quad (4-11)$$

模型的训练及推理方式与 DT 完全一致，相比 DT 模型，StARformer 能够有效的提高模型对图像信息的理解能力，本毕设对上述离线强化学习算法进行了复现<sup>①</sup>，本文的相关复现实验显示相比 DT 更不依赖于初始时目标总奖励  $\hat{R}_0$  的设定，这说明其更倾向于对专家数据的模仿学习，而非与目标总奖励进行对齐，在 5 个不同 Atari 环境下的实验表明，StARformer 的平均得分超过 DT 算法的 30%。

## 4.2 决策模型设计

### 4.2.1 状态空间与动作空间

模型的状态输入由 2 部分构成，分别为  $S^{img}, \mathbf{s}^{card}$ ，其中  $S^{img} \in \mathbb{R}^{18 \times 32 \times 15}$  为单位的网格状特征输入，对于第  $i$  行  $j$  列的特征  $\mathbf{z}_{ij} := (S^{img})_{ij} \in \mathbb{R}^{15}$  表示处于该位置的单位具有如下 4 种特征： $(\mathbf{z}_{ij})_{1:8}$  为类别编码， $(\mathbf{z}_{ij})_9$  为从属派别编码， $(\mathbf{z}_{ij})_{10:12}$  为生命值图像编码， $(\mathbf{z}_{ij})_{13:15}$  为其余条状图像编码； $\mathbf{s}^{card} \in \mathbb{R}^6$  表示当前状态下的两个全局特征： $(\mathbf{s}^{card})_{1:5}$  为当前手牌信息， $(\mathbf{s}^{card})_6$  为当前总圣水量。

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

### 4.2.2 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 5-2 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  $a_i^{delay} = \min\{j - i, T_{delay}\}$ 。

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  $j \in \{t_i, \dots, t_{i+1} - 1\}$  帧作对应的重采样频

<sup>①</sup> <https://github.com/wty-yy/Decision-Transformer-JAX>，复现内容包括：Decision Transformer (DT)，Return-Aligned Decision Transformer (RADT) 和 StARformer，在 Atari 环境中进行了对比试验，并对结果进行可视化。

次为

$$s_j = \max \left\{ \frac{1}{1 - r_a}, \frac{1}{r_a(j - t_i + 1)} \right\}, \quad (t_i \leq j \leq t_{i+1}) \quad (4-12)$$

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 4-1 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。

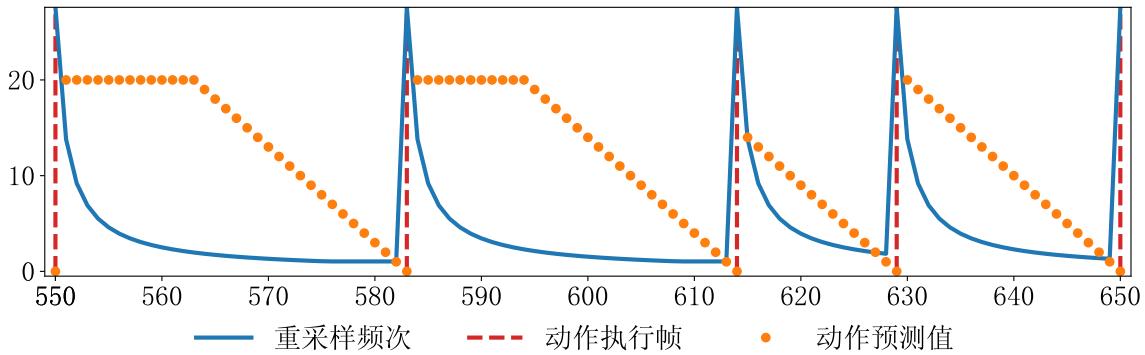


图 4-1 从离线数据集中截取的一段数据, 总共包含 5 个动作帧, 最大间隔帧数阈值  $T_{delay} = 20$ ,

### 4.2.3 模型架构设计

本文设计了 3 种不同的模型架构, 分别基于 StARformer 和 DT 模型, 假设输入的轨迹长度为  $L$ , 时序注意力机制中的输入序列长度  $T$ , 使用  $N$  层 Transformer 堆叠, 则每种模型架构设计细节如下:

**StARformer-3L:** 基于 StARformer 架构, 本文设计的 StARformer-3L 决策模型架构如图 4-2 所示, 其中输入序列长度  $T = 3L$ , 第  $n$  层 Transformer 输出的时序序列记为  $\{z_t^{img_n}, z_t^{card_n}, l_t^{n-1}\}_{t=1}^L$  (序列长度  $3L$ )。右侧时序注意力机制中的因果 Transformer, 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 4-7 中的掩码矩阵  $M_3$ , 其中  $M_{L_0}$  定义为

$$(M_{L_0})_{ij} = \begin{cases} 1, & i = kL_0 - l, j \leq kL_0 \\ 0, & \text{否则} \end{cases}, \quad (k \in \{1, \dots, L\}, l \in \{0, \dots, L_0 - 1\}) \quad (4-13)$$

其本质上是在下三角阵的对角线上顺次放置不交的大小为  $L_0 \times L_0$  的全 1 矩阵。

**StARformer-2L:**  $T = 2L$  的模型架构与 StARformer<sup>[9]</sup> 论文中架构基本一致, 其将图像信息  $s_t^{img}$  与  $s_t^{card}$  编码到同一特征  $z_t$  中, 得到第  $n$  层的 Transformer 输出的时序序列  $\{z_t^n, l_t^{n-1}\}_{t=1}^L$  (序列长度  $2L$ ), 同理需要使用局部交叉注意力, 并将掩码矩阵置为  $M_2$ , 预测中  $a_t^{pos}$  和  $a_t^{select}$  均使用  $z_t$  进行解码得到。

**DT-4L:** 基于 DT 架构, 可以看作仅包含图 4-2 中时序注意力部分, 将时序注意力机制中  $l_t^n$  进行替换, 得到时序序列为  $\{a_{t-1}, R_{t-1}, s_t^{img_n}, s_t^{card_n}\}_{t=1}^L$  (序列长度  $4L$ ), 预

测中  $a_t^{pos}$  和  $a_t^{select}$  分别由  $s_t^{img_n}$  和  $s_t^{card_n}$  对应解码得到。

上述各种模型及不同预测目标的验证比对结果请见表 5-2。

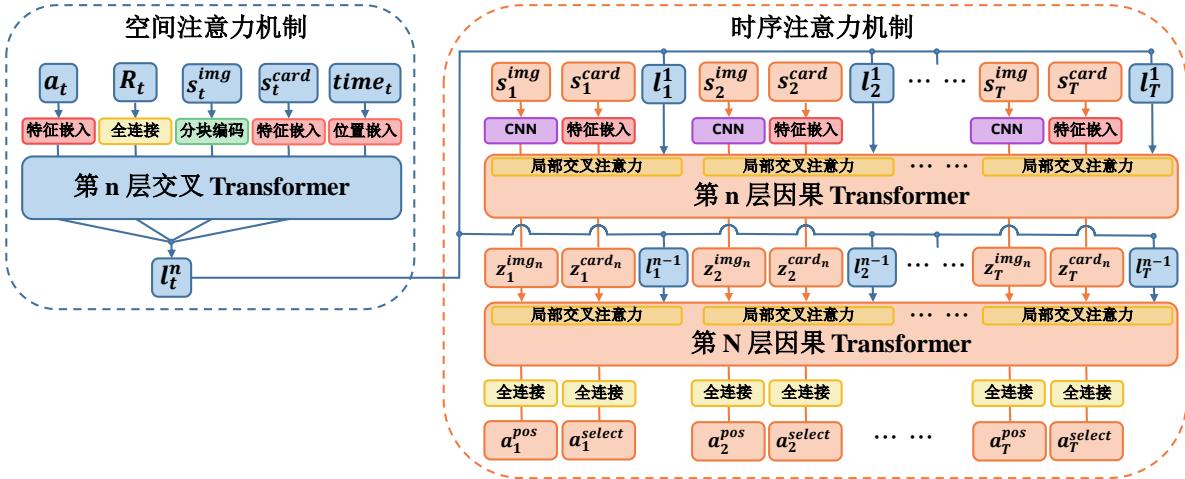


图 4-2 决策模型：基于 StARformer 的 ViT+DT 架构，模型输入为轨迹序列  $(a_t, R_t, s_t)_{t=1}^T$ ，输出为动作预测序列  $(a_t^{pos}, a_t^{select})_{t=1}^T$ 。左侧交叉注意力机制对局部信息  $(a_t, R_t, s_t)$  按空间维度进行编码，并使用 ViT 中分块思路将图像  $s_t^{img}$  转化为序列；右侧因果注意力机制对全局信息  $(s_t^{img}, s_t^{card})$  按时序维度进行编码，并在每层序列输入中引入对应的局部编码信息  $l_t^n$ 。

## 5 数据分析及实验结果

### 5.1 生成式数据集分析

数据集总共分为两部分<sup>①</sup>:

- 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 5-1 所示。
- 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。

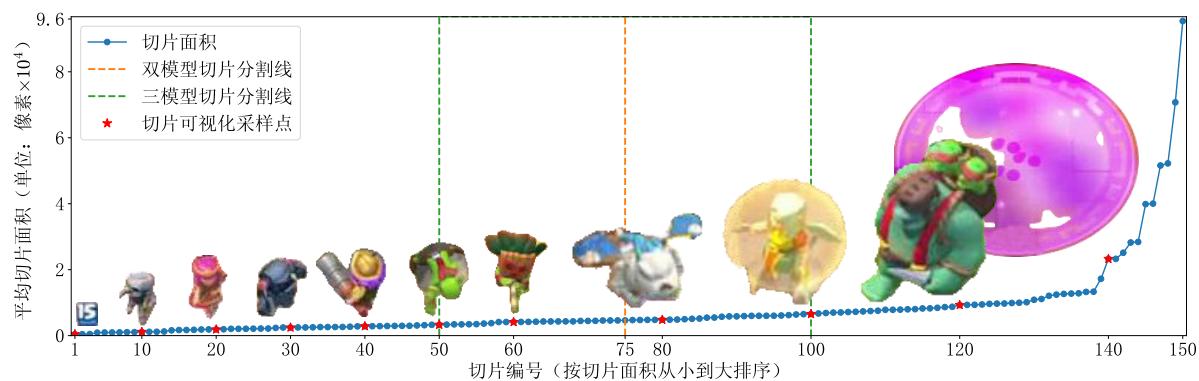


图 5-1 将切片数据集全部切片按平均面积从小到大排序并编号, 从中随机采样出部分切片进行可视化。分别以全体切片面积的二等分和三等分点, 作为双模型和三模型的识别类别分割线。

### 5.2 目标识别模型训练

目标识别模型使用了自己实现的 YOLOv5<sup>②</sup> 和重构后的 YOLOv8 的模型<sup>③</sup>, 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 A-1。

实验结果<sup>④</sup>如表 5-1 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, 1,x 分别对应大与特大型模型, YOLOv8-1×n 表示使用 n 个 YOLOv8-1 模型, 每个子模型分别识别图 5-1 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。

<sup>①</sup> 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>

<sup>②</sup> 复现 YOLOv5 代码: <https://github.com/wty-yy/KataCV/tree/master/katacv/yolov5>

<sup>③</sup> YOLOv8 重构内容: [https://github.com/wty-yy/KataCR/blob/master/assets/yolov8\\_modify.md](https://github.com/wty-yy/KataCR/blob/master/assets/yolov8_modify.md)

<sup>④</sup> 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/YOL0v8>

2. 评测指标：表中 mAP 评测指标表示的是 COCO<sup>[36]</sup> 比赛提出的 COCO mAP 指标，即在 10 种不同 IOU 阈值下计算 PR 曲线下面积求平均得到，AP50、P50 和 R50 分别表示在判断正例的 IOU 阈值为 50% 下的 mAP、平均精度和平均召回率，。

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，做验证机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<sup>[30]</sup> 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，YOLOv8-1 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 5-1 YOLO 模型对比测试结果

模型名称	mAP	AP50	P50	R50	FPS	FPS(T)	mAP(S)	检测器类别数	数据增强
YOLOv5-1	53.2	66.2	84.4	63.8	59	NA	NA	151	
YOLOv8-x	67.7	83.1	<b>93.9</b>	68.3	68	31	39.8	160	
YOLOv8-x	66.8	<b>85.3</b>	90.7	80.4	68	31	35.9	160	✓
YOLOv8-1 × 2	67.4	84.3	89.5	79.8	34	18	43.9	85	✓
YOLOv8-1 × 3	<b>68.8</b>	85.2	89.7	<b>80.9</b>	23	10	<b>48.3</b>	65	✓

### 5.3 决策模型训练

本毕设基于第 3 章中介绍的感知融合技术，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<sup>⑥</sup>，固定双方使用的卡组（具体卡组见附录 A.3），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 4.2.2）。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（定义 2.4），总损失函数如下

$$\mathcal{L} = \sum_{i=1}^N [a_i^{delay} < T_{delay}] [\mathcal{L}_{CE}(\hat{a}_i^{pos}, a_i^{pos}) + \mathcal{L}_{CE}(\hat{a}_i^{select}, a_i^{select}) + \mathcal{L}_{CE}(\hat{a}_i^{delay}, a_i^{delay})] \quad (5-1)$$

其中  $T_{delay}$ ,  $a_i^{pos}$ ,  $a_i^{select}$ ,  $a_i^{delay}$  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（架构设计见 4.2），分别测试了 StARformer 和 DT 架构。

<sup>⑥</sup> 全部专家数据集均已开源：<https://github.com/wty-yy/Clash-Royale-Replay-Dataset>

2. 模型输入的轨迹步长记为  $L$ , 测试了  $L = 30, 50, 100$  的情况。
3. 不同的预测目标 (离散与连续预测见 4.2.2)。
4. 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:

- 重采样: 对稀疏的动作帧进行大量重采样, 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, 当预测当前手牌编号时, 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<sup>⑥</sup>, 模型实时对局的验证结果总结于表5-2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 5-2 决策模型对比

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

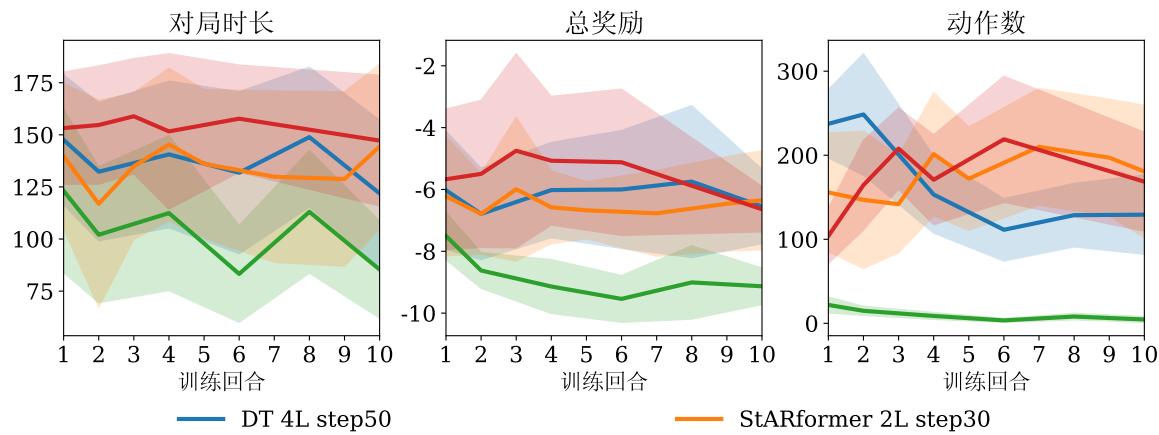
表 5-2 中每个模型取前 10 个训练结果, 将其每次与环境交互 20 个回合得到的最高奖励, 每列的含义分别为:

- 步长  $L$ : 为模型架构设计 4.2.3 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 3-3 进行累计得到的总奖励。

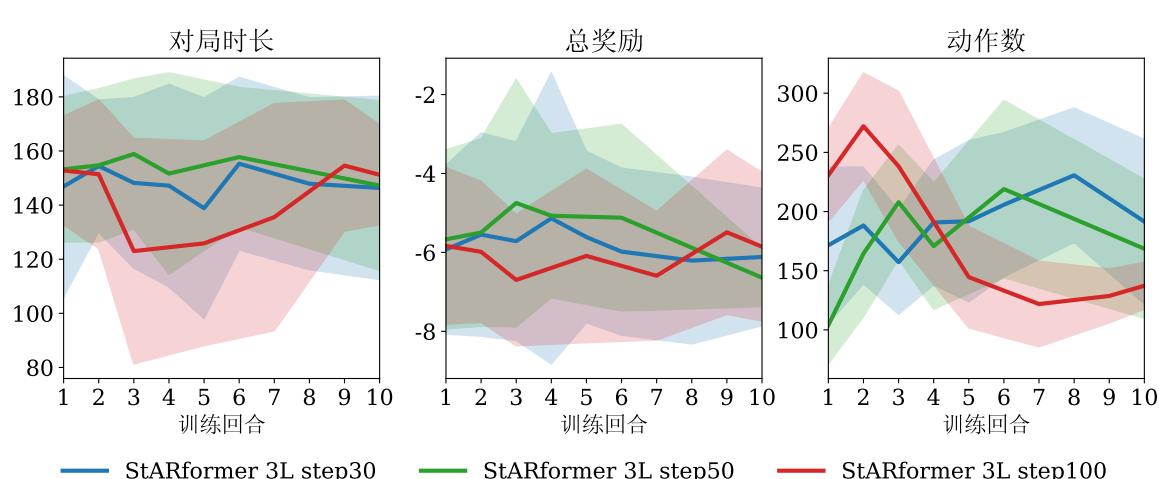
<sup>⑥</sup> 决策模型全部训练曲线: <https://wandb.ai/wty-yy/ClashRoyale%20Policy>

- 对局长：统计每次对局的时长，单位秒。
- 动作数：统计每局智能体成功执行的动作数目。
- 胜率：按照 1.3 中介绍的游戏目标，判定智能体的获胜概率。

图 5-2 中展示了每种模型前 10 个训练结果曲线，从中可以看出，本文将离散预测改为连续预测、StARformer 架构从 2L 修改为 3L 的改动均能够显著提高模型性能，



(a) 不同模型结构



(b) StARformer-3L 采取不同轨迹长度

图 5-2 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。

## 6 总结与展望

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顶尖算法，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。

主要贡献包含以下三点：

1. 数据集制作：本文设计了一种高效制作切片数据集的方法，制作了包含 4654 张切片、共 150 个类别的切片数据集，以及包含 116878 个目标框、共 6939 张图像的目标识别数据集。提出的生成式目标识别数据集算法可以模拟真实对局场景生成带标签的图像，通过生成式图像训练的模型在真实视频流中表现出良好的泛化性，具有很高的识别准确率。
2. 感知融合算法：基于计算机视觉模型的输出结果，设计了感知融合算法，该算法结合视频数据中的上下文信息优化特征结果，进一步提升了识别的准确率。
3. 决策模型改进：在决策模型方面，从架构及预测目标两个方面对传统模型进行改进，将难以学习的离散动作序列转化为连续动作序列，大幅提高了模型性能。制作了包含 105 回合、总共 113981 帧的专家数据集，并基于该离线数据集训练出能够战胜游戏内置 AI 的智能体。

本文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。未来的工作可以从以下几个方面进行扩展：

1. 数据集的扩展与优化：增加数据集的规模和多样性，进一步提升模型的泛化能力和识别准确率。
2. 算法的改进：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。
3. 实际应用的拓展：将本文的方法应用于更多的游戏和实际场景中，验证其通用性和实用价值。

本文的全部代码均已开源，期望能够为相关领域的研究者提供有价值的参考和借鉴。

## 致 谢

感谢西安交通大学数学学院能够给我这次自拟毕设题目的机会，如果错过这次机会，之后可能很难再有充足的时间与精力完成本毕设内容；感谢数学学院强基计划，通过本科课程学习，使我有了扎实的数学基础，能够比较轻松地看懂计算机视觉、强化学习等领域中的论文，理解并对其中的公式进行推导加深理解，最后使用代码进行复现。

感谢兰旭光老师对本毕设的支持，即使本毕设内容充满各种未知挑战，但老师仍给我提供了必要的算力支持，如果没有充足算力，本文进展将相当缓慢，完全无法完成任务。感谢课题组中各位师兄师姐为本次毕设提供思路，其中使用 SAM 辅助切片制作和多目标识别模型进行目标检测思路来自王宇航博士，决策模型中将离散预测转化为连续预测思路来自万里鹏博士，如果没有这些改进，精准的识别模型可能无法实现，决策模型也没有任何性能，战胜游戏中的内置 AI 成为幻想。还要感谢我身边朋友的支持，他们的支持使我愈发坚定将不可能变为可能的决心。

感谢我的父母对我一直以来的支持，使我在高中参加算法竞赛，打下扎实的编程基础，并支持我去追随自己的儿时的梦想——“做出一个能够与现实进行直接交互并改善生活的智能体”。如果没有这样坚定的梦想，我可能早在制作数据集的枯燥过程中放弃，但通过大学的各种知识与技术的学习，使我看到了前进的道路，坚定下自己的信念，将全部代码从零实现了出来，从而初步完成了本毕设开题时所设定的目标，也成功地走出了第一步。

## 参考文献

- [1] 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.
- [2] Silver D, Hubert T, Schrittwieser J, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm[J]. *arXiv preprint arXiv:1712.01815*, 2017.
- [3] Berner C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning[J]. *arXiv preprint arXiv:1912.06680*, 2019.
- [4] Hochreiter S, Schmidhuber J. Long short-term memory[J]. *Neural computation*, 1997, 9(8):1735-1780.
- [5] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. *arXiv preprint arXiv:1707.06347*, 2017.
- [6] 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.
- [7] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. *Nature*, 2015, 518(7540):529-533.
- [8] 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.
- [9] 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.
- [10] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. *Proceedings of the IEEE*, 1998, 86(11):2278-2324.
- [11] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database[C]//2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009: 248-255.
- [12] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. *Communications of the ACM*, 2017, 60(6):84-90.
- [13] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. *arXiv preprint arXiv:1409.1556*, 2014.
- [14] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
- [15] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
- [16] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.
- [17] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. *arXiv preprint arXiv:2010.11929*, 2020.
- [18] Ding X, Zhang X, Ma N, et al. Repvgg: Making vgg-style convnets great again[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 13733-13742.
- [19] Yu W, Luo M, Zhou P, et al. Metaformer is actually what you need for vision[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 10819-10829.
- [20] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. *Advances in neural information processing systems*, 2017, 30.

- [21] 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.
- [22] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer, 2016: 21-37.
- [23] Carion N, Massa F, Synnaeve G, et al. End-to-end object detection with transformers[C]//European conference on computer vision. Springer, 2020: 213-229.
- [24] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.
- [25] Shi B, Bai X, Yao C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 39(11):2298-2304.
- [26] Graves A, Fernández S, Gomez F, et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks[C]//Proceedings of the 23rd international conference on Machine learning. 2006: 369-376.
- [27] Huang Z, Xu W, Yu K. Bidirectional lstm-crf models for sequence tagging[J]. arXiv preprint arXiv:1508.01991, 2015.
- [28] Du Y, Li C, Guo R, et al. Pp-ocr: A practical ultra lightweight ocr system[J]. arXiv preprint arXiv:2009.09941, 2020.
- [29] Jocher G, Chaurasia A, Qiu J. Ultralytics YOLO[CP/OL]. 2023. <https://github.com/ultralytics/ultralytics>.
- [30] Zhang Y, Sun P, Jiang Y, et al. Bytetrack: Multi-object tracking by associating every detection box [C]//European conference on computer vision. Springer, 2022: 1-21.
- [31] Kirillov A, Mintun E, Ravi N, et al. Segment anything[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 4015-4026.
- [32] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.
- [33] Everingham M, Van Gool L, Williams C K, et al. The pascal visual object classes (voc) challenge[J]. International journal of computer vision, 2010, 88:303-338.
- [34] Zhang Z, He T, Zhang H, et al. Bag of freebies for training object detection neural networks[J]. arXiv preprint arXiv:1902.04103, 2019.
- [35] 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.
- [36] Lin T Y, Maire M, Belongie S, et al. Microsoft coco: Common objects in context[C]//Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. Springer, 2014: 740-755.
- [37] Jocher G. YOLOv5 by Ultralytics[CP/OL]. 2020. <https://github.com/ultralytics/yolov5>. DOI: 10.5281/zenodo.3908559.
- [38] Bochkovskiy A, Wang C Y, Liao H Y M. Yolov4: Optimal speed and accuracy of object detection[J]. arXiv preprint arXiv:2004.10934, 2020.
- [39] Wang C Y, Liao H Y M, Wu Y H, et al. Cspnet: A new backbone that can enhance learning capability of cnn[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020: 390-391.

- [40] He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2015, 37(9):1904-1916.
- [41] Liu S, Qi L, Qin H, et al. Path aggregation network for instance segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 8759-8768.
- [42] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.
- [43] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.
- [44] Liao M, Wan Z, Yao C, et al. Real-time scene text detection with differentiable binarization[C]// Proceedings of the AAAI conference on artificial intelligence: volume 34. 2020: 11474-11481.
- [45] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International conference on machine learning. pmlr, 2015: 448-456.
- [46] Levenshtein V I, et al. Binary codes capable of correcting deletions, insertions, and reversals[C]// Soviet physics doklady: volume 10. Soviet Union, 1966: 707-710.
- [47] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. 2018.
- [48] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

## 附录 A 相关内容补充

### A.1 动态概率分布

动态概率分布用于切片的中心点位选取，可以将生成的单位形成团簇，产生更多重叠部分，能有效提高识别数据的多样性，具体实现过程如下，设当前待选取点位为离散的  $W \times H$  矩阵，切片的待选二维离散分布为  $D : \{1, 2, \dots, W\} \times \{1, 2, \dots, H\} \rightarrow \mathbb{R}$ ，动态修改范围为  $S = 3$ ，令  $(i, j) \sim D$ ，则修改  $D$  中以  $(i, j)$  为中心附近大小为  $S \times S$  的分布：

$$\begin{bmatrix} d_{i-1,j-1} & d_{i-1,j} & d_{i-1,j+1} \\ d_{i,j-1} & d_{ij} & d_{i,j+1} \\ d_{i+1,j-1} & d_{i+1,j} & d_{i+1,j+1} \end{bmatrix} \rightarrow \frac{1}{f(d_{ij})} \begin{bmatrix} d_{i-1,j-1} & d_{i-1,j} & d_{i-1,j+1} \\ d_{i,j-1} & \frac{f(d_{ij})}{2}d_{ij} & d_{i,j+1} \\ d_{i+1,j-1} & d_{i+1,j} & d_{i+1,j+1} \end{bmatrix} \quad (\text{附录 A-1})$$

其中  $f(d_{ij}) = \frac{d_{ij}}{2 \sum_{u,v=1}^S [d_{i+u,j+v} \neq 0]}$ 。做法是使中心概率下降  $d_{ij}/2$ ，并将周围的非零概率点位平均分配  $d_{ij}/2$ 。

### A.2 数据增强

表 A-1 数据增强参数范围

数据增强类型	参数变换范围	单位
HSV 增强	$\pm(0.015, 0.7, 0.4)$	比例系数
图像旋转	$(-5, 5)$	度
横纵向随机平移	$(-0.05, 0.05)$	比例系数
图像缩放	$(-0.2, 0.2)$	比例系数
图像左右反转	0.5	概率大小

### A.3 模型测试卡组

表 A-2 我方卡组 (平均圣水花费 2.6)

卡牌名称	类型	攻击目标	圣水花费
骷髅兵	部队	地面	1
冰雪精灵	部队	地面和空中	1
滚木	法术	地面	2
戈仑冰人	部队	建筑	2
加农炮	建筑	地面	3
火枪手	部队	地面和空中	4
野猪骑士	部队	建筑	4
火球	法术	地面和空中	4

表 A-3 8000 分内置 AI 敌方卡组 (平均圣水花费 4.6)

卡牌名称	类型	攻击目标	圣水花费
野蛮人滚筒	法术	地面	2
飓风法术	法术	地面和空中	3
狂暴樵夫	部队	地面	4
骷髅飞龙	部队	地面和空中	4
野蛮人	部队	地面	5
雷电飞龙	部队	地面和空中	5
圣水收集器	建筑	无	6
戈仑石人	部队	建筑	8

注：测试时间为 2024 年 5 月，第 59 赛季。

## 附录 B 论文相关代码

本论文全部代码均已开源 <https://github.com/wty-yy/katacr>，总计 1.4 万行左右，下面将展示论文中提到的部分代码块。

### B.1 生成式数据集

---

```

1  """
2  完整代码: https://github.com/wty-yy/KataCR/blob/master/katacr/build\_dataset/generator.py
3  """
4  ... # import packages, define constants
5
6  class Generator:
7      def __init__(self,
8          background_index: int | None = None,
9          unit_list: Tuple[Unit, ...] = None,
10         seed: int | None = None,
11         intersect_ratio_thre: float = 0.5,
12         map_update: dict = {'mode': 'naive', 'size': 5},
13         augment: bool = True,
14         dynamic_unit: bool = True,
15         avail_names: Sequence[str] = None,
16         noise_unit_ratio: float = 0.0,
17     ):
18         """
19         Args:
20             background_index: Use image file name in
21             `dataset/images/segment/backgrounds/background{index}.jpg` as current background.
22             unit_list: The list of units will be generated in area.
23             seed: The random seed.
24             intersect_ratio_thre: The threshold to filter overlapping units.
25             map_update_size: The changing size of dynamic generation distribution (SxS).
26             augment: If toggled, the mask augmentation will be used.
27             dynamic_unit: If toggled, the frequency of each unit will tend to average.
28             avail_names: Specify the generation classes.
29             noise_unit_ratio: The ratio of unavailable unit (noise unit) in whole units.
30         Variables:
31             map_cfg (dict):
32                 'ground': The 0/1 ground unit map in `katacr/build_dataset/generation_config.py`.
33                 'fly': The 0/1 fly unit map in `katacr/build_dataset/generation_config.py`.
34                 'update_size': The size of round square.
35             ...
36
37     def build(self, save_path="", verbose=False, show_box=False, box_format='cxcywh',
38             img_size=None):
39         ... # Build image and bounding box
40
41     def add_tower(self, king=True, queen=True):
42         ... # Add king and queen tower
43
44     def add_unit(self, n=1):
45         ... # Add unit in [ground, flying, others] randomly. Unit list looks at
46             `katacr/constants/label_list.py`
```

---

```

47     ... # Reset generator
48
49 if __name__ == '__main__':
50     generator = Generator(seed=42, background_index=25, intersect_ratio_thre=0.5, augment=True,
51     ↪ map_update={'mode': 'naive', 'size': 5}, avail_names=None)
52     for i in range(10):
53         generator.add_tower()
54         generator.add_unit(n=40)
55         x, box, _ = generator.build(verbose=False, show_box=True)
56         generator.reset()

```

---

## B.2 特征融合

### B.2.1 状态特征提取部分代码

---

```

1 """
2 完整代码:
3     ↪ https://github.com/wty-yy/KataCR/blob/master/katacr/policy/perceptron/state_builder.py
4 """
5
6 ... # import packages, define constants
7
8 class StateBuilder:
9     def get_state(self, verbose=False):
10         ... # Build state
11
12     def update(self, info: dict, deploy_cards: set):
13         """
14             Args:
15                 info (dict): The return in `VisualFusion.process()`,
16                     which has keys=[time, arena, cards, elixir]
17                 deploy_cards (set): Get deploy_cards from action_builder.
18         """
19
20         self.time: int = info['time'] if not np.isinf(info['time']) else self.time
21         self.arena: CRResults = info['arena']
22         self.cards: List[str] = info['cards']
23         self.elixir: int = info['elixir']
24         self.card2idx: dict = info['card2idx']
25         self.parts_pos: np.ndarray = info['parts_pos'] # shape=(3, 4), part1,2,3, (x,y,w,h)
26         self.box = self.arena.get_data() # xyxy, track_id, conf, cls, bel
27         self.img = self.arena.get_rgb()
28         self.frame_count += 1
29         ### Step 0: Update belong memory ###
30         self._update_bel_memory()
31         ### Step 1: Find text information ###
32         self._find_text_info(deploy_cards)
33         ### Step 2: Build bar items ###
34         self._build_bar_items()
35         ### Step 3: Combine units and bar2 with their BarItem ###
36         self._combine_bar_items()
37         ### Step 4: Update bar history ###
38         self._update_bar_items_history()
39         ### Step 5: Update class memory, if body exists ###
40         self._update_cls_memory()

```

---

## B.2.2 动作特征提取部分代码

---

```

1  """
2  完整代码:
3      ↳ https://github.com/wty-yy/KataCR/blob/master/katacr/policy/perceptron/action_builder.py
4  """
5  ... # import packages, define constants
6
7  class ActionBuilder:
8      ... # define other functions
9
10 def get_action(self, verbose=False):
11     ... # Get from actions queue
12
13 def update(self, info):
14     """
15     Args:
16         info (dict): The return in `VisualFusion.process()`,
17             which has keys=[time, arena, cards, elixir]
18     """
19     self.time: int = info['time'] if not np.isinf(info['time']) else self.time
20     self.arena: CRRResults = info['arena']
21     self.cards: dict = info['cards']
22     self.elixir: int = info['elixir']
23     self.card2idx: dict = info['card2idx']
24     self.box = self.arena.get_data() # xyxy, track_id, conf, cls, bel
25     self.img = self.arena.get_rgb()
26     self.frame_count += 1
27     ### Step 1: Update elixir history ###
28     self._update_elixir()
29     # print("Elixirs:", self.elixirs)
30     ### Step 2: Update card memory ###
31     self._update_cards()
32     ### Step 3: Update last elixir ###
33     self._update_mutation_elixir_num()
34     ### Step 4: Find new action ###
35     self._find_action()

```

---

## B.2.3 奖励特征提取部分代码

---

```

1  """
2  完整代码:
3      ↳ https://github.com/wty-yy/KataCR/blob/master/katacr/policy/perceptron/reward_builder.py
4  """
5  ... # import packages, define constants
6
7  class RewardBuilder:
8      ... # Define other functions
9
10 def get_reward(self, verbose=False):
11     ### Update King Tower ###
12     ### Update Tower ###
13     ### Calculate Reward ###
14     ...
15
16 def update(self, info):
17     """
18     Args:

```

---

```
18     info (dict): The return in `VisualFusion.process()`,  
19         which has keys=[time, arena, cards, elixir]  
20     """  
21     self.time: int = info['time'] if not np.isinf(info['time']) else self.time  
22     self.arena: CRResults = info['arena']  
23     self.elixir: int = info['elixir']  
24     self.img = self.arena.get_rgb()  
25     self.box = self.arena.get_data() # xyxy, track_id, conf, cls, bel  
26     self.frame_count += 1
```

---