强化学习大作业报告

潘昱锜
中国科学院自动化研究所
202418014628018
panyuqi2024@ia.ac.cn

温尧智
中国科学院自动化研究所
2024180204280028
wenyaozhi2024@ia.ac.cn

谷绍伟
中国科学院自动化研究所
202418020428007
gushaowei2024@ia.ac.cn

Abstract

本报告针对实时格斗游戏场景，基于 FightingICE 平台开展强化学习智能体训练任务，目标是使 AI 在与固定对手 “MctsAI” 的对战中实现策略优化。通过状态空间建模、动作策略优化及多场景测试，验证算法在高维博弈环境中的有效性。实验设计并对比了 Sarsa、Q-Learning 和 DQN 三种经典强化学习算法，其中 Sarsa 与 DQN 进行 300 轮训练，Q-Learning 进行 500 轮训练。多场景测试表明：Sarsa 与 DQN 在 100 局标准对战中分别取得 53.0% 和 56.0% 胜率，在限制血量场景下胜率进一步提升至 75.0% 和 70.0%；Q-Learning 标准胜率仅 18.0%，限制血量场景胜率 44.0%。结果表明，Sarsa 与 DQN 能有效应对高维博弈环境，其中 DQN 泛化性更优，Q-Learning 因状态空间限制未能充分收敛。

Keywords 强化学习 · 格斗游戏 · DQN · Sarsa · Q-Learning

1 任务介绍

拳皇（Fighting Game）作为经典的格斗类电子游戏，玩家需操控角色进行 1 对 1 对战，通过连招、格挡、闪避等操作击败对手，其对抗性与策略性深受玩家喜爱，如图 1 所示。本次实验基于 Fighting Game 的实时格斗场景，目标是通过强化学习方法设计并训练一个具备智能行为的游戏 AI，使其在与固定对手 “MctsAi” 的对抗中占据优势。

本实验基于专为格斗游戏 AI 研究设计的 FightingICE 平台展开。该平台能够按照一定帧率实时返回游戏状态信息，并提供可交互、可训练的格斗环境，允许玩家或 AI 在限定时间内通过连续动作打击对手获取胜利。游戏采用 1 对 1 对抗形式，我方 AI 控制一名格斗角色，与系统内置的 MctsAi 进行

单场战斗。由于对抗过程属于实时同步博弈，AI 需在信息有限且对手行为不完全可预知的条件下选择动作。同时，游戏设定双方初始血量和能量固定，战斗目标为在有限时间内尽可能削减对手血量或率先将其击败。我们将该任务建模为强化学习问题，智能体控制的角色在训练中根据决策策略不断尝试各种动作，从环境交互中获得伤害、位置改变、攻击情况等反馈，并根据反馈信息优化决策策略，从而逐步学习有效的对战策略，达到最大化击中对手概率、避免被击中、赢得比赛的训练目标。

本实验实现并对比了 Sarsa、Q-Learning 和 DQN（Deep Q-Network）三种经典强化学习算法。这些算法涵盖从表格型到深度学习增强的强化学习方法，能够从不同层面应对格斗游戏中庞大的状态空间与复杂的策略博弈挑战。最终，在训练完成后，将使用所学策略与 MctsAi 进行 100 局对战，通过统计胜率与双方血量差的平均值，量化各算法训练出的 AI 性能。

图 1: 格斗游戏

2 任务分工

本次实验任务由三名成员共同完成，具体分工如表1：

表 1: 任务分工表

姓名	负责任务
潘昱锜	Sarsa 算法代码实现、训练环境搭建、实验数据处理和结果分析、实验报告撰写
温尧智	Q-Learning 算法代码实现、训练环境搭建和模型训练、模型测试、实验报告撰写
谷绍伟	DQN 算法代码实现、训练环境搭建、模型测试、实验数据处理、实验报告撰写

3 实现算法

我们分别实现了 Sarsa、Q-Learning 和 DQN 的强化学习算法，并分别用这三种方法对智能体进行了训练。以下将详细介绍三种算法的原理和实现细节。

3.1 Q-Learning

Q-learning 是一种经典的无模型强化学习算法，通过迭代优化状态 - 动作价值函数（Q 函数）实现最优策略的自主学习。算法的核心思想是通过学习状态-动作对的 Q 值，来评估在每个状态下采取不同动作的优劣。智能体在与环境交互的过程中，根据当前的 Q 值和获得的奖励，逐步更新 Q 值，最终收敛到最优的 Q 函数 Q*。在每一步中，智能体根据一定的策略（如 ϵ-贪婪策略）选择动作，然后根据更新规则来调整 Q 值。

Q-learning 的更新规则为：

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

其中，s 和 a 分别是当前的状态和动作，s' 是执行动作 a 后的新状态，r 是获得的奖励，α 是学习率 ($0 < \alpha \leq 1$),γ 是折扣因子 ($0 \leq \gamma < 1$) 。$\max_{a'} Q(s',a')$ 表示在新状态 s' 下所有可能动作的最大 Q 值，它反映了智能体对未来收益的估计。

为了平衡探索和利用，Q-learning 通常采用 ϵ-贪婪策略。具体来说，智能体以概率 ϵ 随机选择一个动作，以概率 $1-\epsilon$ 选择当前状态下 Q 值最大的动作。这种策略既保证了智能体能够探索未知的环境，又能够在一定程度上利用已知的最优动作。表示如下：

$$\begin{cases} \text{随机选择一个动作} & \text{概率为}\epsilon, \\ \arg\max_a Q(s_t,a) & \text{概率为}1-\epsilon. \end{cases}$$

Q-learning 算法在满足一定条件下可以收敛到最优的 Q 函数 Q^* 。具体来说，需要满足以下条件：

1. 每个状态-动作对 (s,a) 被无限次地访问。
2. 学习率 α 满足：

$$\sum_t \alpha_t(s,a) = \infty \quad \text{和} \quad \sum_t \alpha_t^2(s,a) < \infty$$

这意味着学习率在迭代过程中逐渐减小，但总和趋于无穷大，以保证算法能够逐步收敛。Q-learning 算法流程如算法1所示。

3.2 Sarsa

Sarsa（State-Action-Reward-State-Action）是一种基于时序差分（Temporal-Difference, TD）方法的强化学习算法，属于 on-policy（依赖于当前策略）的值迭代方法。其核心思想是在智能体与环境交互的过程中，通过估计当前策略下某一状态-动作对的价值（Q 值），逐步更新并改进策略，从而实现学习最优行为，其算法流程如算法2。

Algorithm 1 Q-learning 算法

Require: 状态空间 $\mathcal{S}$, 动作空间 $\mathcal{A}$, 学习率 $\alpha \in (0,1]$, 折扣因子 $\gamma \in [0,1)$, 探索率 $\epsilon \in [0,1]$, 最大迭代次数 T

Ensure: 最优动作价值函数 $Q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

初始化 $Q(s,a) \leftarrow 0$ 或随机小值, 对于所有 $s \in \mathcal{S}, a \in \mathcal{A}$

for $t = 1$ to T **do**

重置环境, 获取初始状态 s_1

$done \leftarrow$ false

while $\neg done$ **do**

动作选择:

if 随机数 $< \epsilon$ **then**

选择随机动作 $a_t \in \mathcal{A}$

else

选择 $a_t \leftarrow \arg\max_a Q(s_t, a)$

end if

环境交互:

执行动作 a_t, 观察奖励 r_t 和下一状态 s_{t+1}

Q 值更新:

$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$

状态转换:

$s_t \leftarrow s_{t+1}$

检查是否达到终止条件，更新 $done$

end while

探索率衰减 (可选): $\epsilon \leftarrow \epsilon \times 0.99$

end for

return Q^*

在每一步交互中，Sarsa 算法会采集五元组（当前状态 s、当前动作 a、即时奖励 r、下一状态 s' 和下一动作 a'），并使用如下更新公式对 Q 值进行迭代：

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{1}$$

其中，α 是学习率，γ 是折扣因子。该更新规则说明，Sarsa 通过比较当前 Q 值与基于“下一状态和下一动作”的估计回报之间的差距来调整当前策略。

与 Q-Learning 不同，Sarsa 是策略一致的（policy-consistent）：它评估和改进的策略是相同的，也就是说，它在学习过程中使用与实际行为相同的策略进行更新（如 ϵ-greedy 策略）。这种一致性使得 Sarsa 在某些环境中表现得更加稳健，尤其是在具有高风险或不确定性的环境中，因为它更容易规避那些在理论上可能收益高但在当前策略下不太可能执行的动作。

在格斗游戏任务中，Sarsa 的优势体现在能够在训练阶段通过持续的试错积累经验，并不断调整对当前策略下的动作价值估计，逐步形成一套能够在实时对抗中稳定发挥的策略。在本任务中，格斗环境

的状态由连续与离散信息共同构成，用 $s \in \mathbb{R}^n$ 表示，状态空间高维且不适合直接离散建表。因此，我们采用线性函数逼近来近似 Q 值函数。具体地，定义一个线性映射 $\mathcal{Q}(s) = w \cdot s$，其中 $w \in \mathbb{R}^{|A| \times n}$ 是可学习的参数，$|A|$ 为动作空间大小。每个动作对应一组权重，策略采用 ϵ-greedy 贪心选择动作：

$$a = \arg\max \mathcal{Q}(s) \tag{2}$$

Sarsa 的更新公式在此基础上改写为：

$$w_{a_t} = w_{a_t} + \alpha \cdot \big(r + \gamma \cdot \mathcal{Q}_{a_{t+1}}(s_{t+1}) - \mathcal{Q}_{a_t}(s_t)\big) \cdot s_t \tag{3}$$

通过不断更新权重，模型能够逐步学习在复杂状态下执行有效动作的策略，从而在对战中提升表现。

Algorithm 2 Sarsa 算法（线性函数逼近）

Require: 状态空间 $\mathcal{S}$, 动作空间 $\mathcal{A}$, 学习率 $\alpha \in (0, 1]$, 折扣因子 $\gamma \in [0, 1)$, 探索率 $\epsilon \in [0, 1]$, 最大迭代次数 T

Ensure: 近似动作价值函数参数 $w \in \mathbb{R}^{|\mathcal{A}| \times n}$

初始化参数 $w \leftarrow$ 小随机值

for $t = 1$ to T **do**

重置环境, 获取初始状态 s_1

使用 ϵ-greedy 策略选择初始动作 a_1

$done \leftarrow$ false

while $\neg done$ **do**

执行动作 a_t, 观察奖励 r_t 和下一状态 s_{t+1}

使用 ϵ-greedy 策略从 s_{t+1} 选择下一个动作 a_{t+1}

计算当前 Q 值：$\mathcal{Q}_{a_t}(s_t) = w_{a_t} \cdot s_t$

计算目标 Q 值：$\mathcal{Q}_{a_{t+1}}(s_{t+1}) = w_{a_{t+1}} \cdot s_{t+1}$

权重更新:

$w_{a_t} \leftarrow w_{a_t} + \alpha \cdot \big(r_t + \gamma \cdot \mathcal{Q}_{a_{t+1}}(s_{t+1}) - \mathcal{Q}_{a_t}(s_t)\big) \cdot s_t$

$s_t \leftarrow s_{t+1}, \quad a_t \leftarrow a_{t+1}$

检查是否达到终止条件，更新 $done$

end while

探索率衰减 (可选): $\epsilon \leftarrow \epsilon \times 0.9995$

end for

return w

3.3 DQN

DQN（Deep Q-Network）是一种 off policy 的强化学习算法，结合了深度学习和 Q-learning 的思想。它通过使用深度神经网络来近似 Q 值函数，从而能够处理高维状态空间和复杂的输入数据。DQN 引入了两个关键技术：经验回放（Experience Replay）和目标网络（Target Network）。经验回放通过存储智能体的经验（状态、动作、奖励、下一状态）并从中随机采样来打破数据相关性，提高训练稳定性。目标网络则用于计算目标 Q 值，通过定期从主网络复制参数，减少 Q 值更新的震荡，其算法流程如算法3。

Algorithm 3 DQN 算法流程

Require: 学习率 α, 折扣因子 γ, 初始探索率 ϵ, 经验回放缓冲区大小 M_{size}, 批次大小 B, 目标网络更新频率 C

Ensure: 优化的 Q 网络 Q_θ

初始化本地 Q 网络 Q_θ 和目标 Q 网络 $Q_{\hat{\theta}}$，令 $\hat{\theta} \leftarrow \theta$

初始化经验回放缓冲区 $\mathcal{D}$

for 每个训练回合 **do**

重置 FightingICE 环境，获取初始状态 s_t

while 回合未结束 **do**

动作选择: 以概率 ϵ 随机选择动作 a_t，否则 $a_t \leftarrow \arg\max_a Q_\theta(s_t, a)$

执行动作 a_t, 观察奖励 r_t, 下一状态 s_{t+1}, 以及是否终止 $done$

将 $(s_t, a_t, r_t, s_{t+1}, done)$ 存入 $\mathcal{D}$

$s_t \leftarrow s_{t+1}$

if 缓冲区 $\mathcal{D}$ 中经验数量足够 **then**

从 $\mathcal{D}$ 中随机采样 B 个经验 $(s_j, a_j, r_j, s'_j, done_j)$

计算目标 Q 值 y_j:

if $done_j$ **then**

$y_j \leftarrow r_j$

else

$y_j \leftarrow r_j + \gamma \max_{a'} Q_{\hat{\theta}}(s'_j, a')$

end if

计算损失 $L = \frac{1}{B}\sum_j (y_j - Q_\theta(s_j, a_j))^2$

更新网络参数 θ 以最小化 L

end if

定期更新目标网络参数: $\hat{\theta} \leftarrow \theta$ (例如每 C 步)

衰减 ϵ

end while

end for

return Q_θ

DQN 的核心思想是使用深度神经网络 $Q(s, a; \theta)$ 来近似最优动作价值函数 $Q^*(s, a)$，其中 θ 表示网络参数。与传统 Q-learning 不同，DQN 不再维护状态-动作对的 Q 值表格，而是通过神经网络的非线性映射能力学习状态到 Q 值的复杂关系。网络的输入为状态向量 s，输出为所有动作对应的 Q 值向量 $[Q(s, a_1; \theta), Q(s, a_2; \theta), ..., Q(s, a_{|A|}; \theta)]$。

DQN 的损失函数采用均方误差形式，定义为:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[(y - Q(s, a; \theta))^2\right] \tag{4}$$

其中，y 为目标 Q 值，$\mathcal{D}$ 为经验回放缓冲区。目标 Q 值的计算公式为:

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-) \tag{5}$$

这里 θ^- 表示目标网络的参数，它通过定期复制主网络参数获得。

经验回放机制将智能体的历史经验 (s_t, a_t, r_t, s_{t+1}) 存储在容量为 N 的缓冲区中。训练时从缓冲区中随机采样小批量数据进行梯度更新，这样做有两个关键优势：（1）打破了连续经验间的时序相关性，使训练数据更接近独立同分布假设；（2）提高了数据利用效率，每个经验可以被多次使用。

目标网络技术通过维护两个结构相同但参数不同的神经网络来稳定训练过程。主网络 $Q(s, a; \theta)$ 用于动作选择和参数更新，目标网络 $Q(s, a; \theta^-)$ 专门用于计算目标 Q 值。目标网络参数每隔 C 步从主网络复制一次：$\theta^- \leftarrow \theta$，这种" 软更新" 机制有效减少了目标值的震荡，提升了学习稳定性。

在格斗游戏任务中，DQN 通过深度网络的表征学习能力，能够从高维状态特征中自动提取关键信息，学习复杂的对战策略。相比于线性函数逼近方法，DQN 具有更强的非线性建模能力，能够捕捉状态间的复杂关系和隐含模式，从而在面对复杂博弈环境时展现出更好的泛化性能。

4 实验结果

4.1 实验环境

模型的训练与测试都在 FightingICE 平台进行，该平台使用 java 语言开发并通过 python 调用。由于在服务器端使用 FightingICE 需配置虚拟桌面且 python 和 java 程序间的通信容易中断，我们选择在本地 Windows 电脑上进行实验。

4.2 训练

在本任务中，我们基于 FightingICE 平台，以 ZEN 角色为学习主体，分别实现并训练了三种强化学习算法：Sarsa、Q-Learning 和 DQN。对于 Sarsa 和 DQN 算法，每个智能体与环境交互训练 300 个 episode；对于 Q-Learning，我们使其与环境交互训练 500 个 episode。

对于 Sarsa 与 Q-Learning 算法，我们使用线性函数逼近器对状态-动作价值函数（Q 函数）进行建模。具体地，Q 值函数 $\hat{Q}(s, a)$ 表示为动作索引 a 对应的线性权重向量 $\mathbf{w}_a$ 与状态特征向量 $\mathbf{s}$ 的点积：$\hat{Q}(s, a) = \mathbf{w}_a^\top \mathbf{s}$。状态向量维度为 $n = 144$，动作空间大小为 $|A| = 40$，权重矩阵初始化为零矩阵，采用 ϵ-greedy 策略进行探索（初始 $\epsilon = 1.0$，训练过程中按比例逐渐衰减，直至预设下限）。学习率设置为 $\alpha = 0.01$，折扣因子为 $\gamma = 0.95$。

在 DQN 算法的实现中，我们构建了一个三层全连接神经网络作为 Q 值函数逼近器（Q-Network），该网络两个隐藏层激活函数均为 ReLU。我们同时维护一个目标网络（Target Network）用于稳定 Q 值的估计，每经过固定步数对其进行参数同步。训练过程中，智能体通过 ϵ-greedy 策略进行动作选择，初始 $\epsilon = 1.0$，每步按速率进行衰减。经验采用回放机制进行采样，经验池容量为 10000，batch size 为 32。优化器选用 Adam，学习率为 0.001，折扣因子设置为 $\gamma = 0.95$。学习过程中，目标 Q 值由目标网络估计最大 Q 值并构造 TD 目标，损失函数采用均方误差（MSE）。

在训练过程中三种算法的血量差变化曲线和 Reward 变化曲线分别如图 2 和图 3 所示。可以观察到，三种算法均表现出一定程度的收敛趋势，随着训练的进行，三者的血量差和 Reward 均逐步趋于稳定，说明智能体策略逐渐学会有效规避对手攻击并最大化伤害输出。

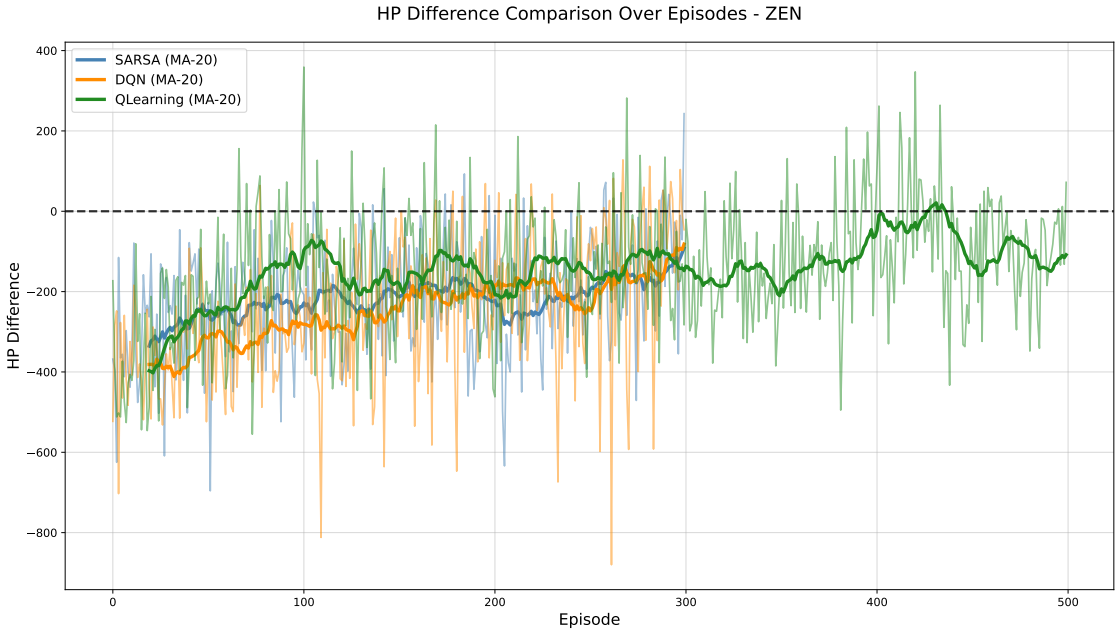

图 2: 三种算法在训练过程中的血量差变化曲线（带滑动平均）

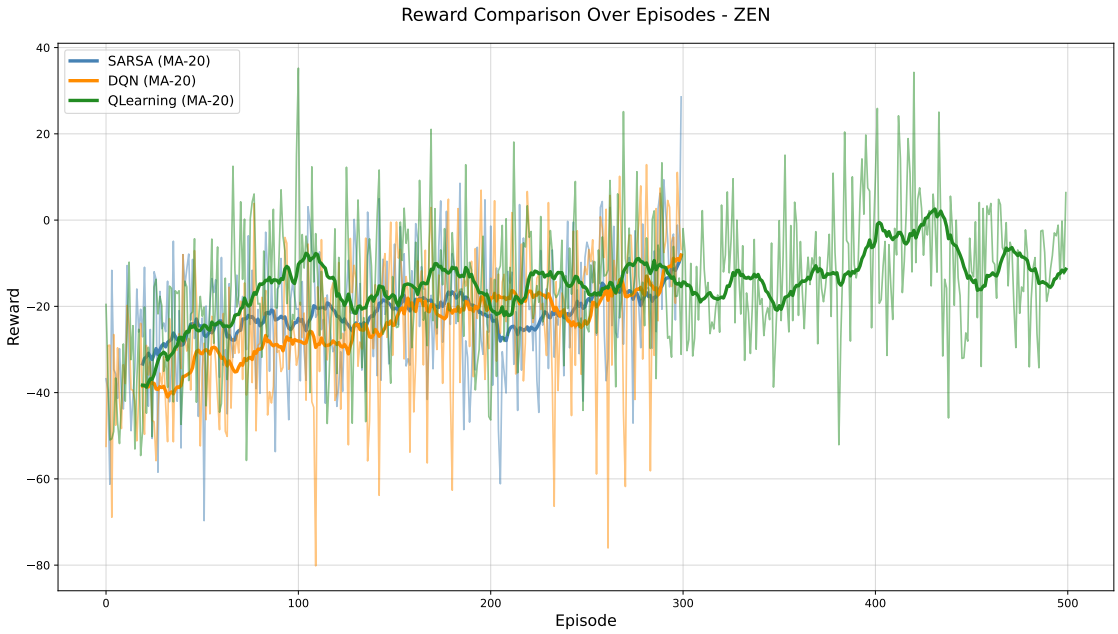

图 3: 三种算法在训练过程中的 Reward 变化曲线（带滑动平均）

4.3 测试

在模型训练完成后，我们对每种算法分别进行了 100 次独立对战测试，并统计了血量差的分布（图 4）、平均血量差以及胜率等指标，用于评估其在未见对局中的泛化能力和实战表现。从结果来看，SARSA 以平均血量差 15.7 和 53.0% 的胜率展现出较为平衡且稳定的对战性能；DQN 虽平均血量差略低（14.3），但胜率达到 56.0%，为三者中最高，说明其策略更具实战效果；相比之下，Q-Learning

的平均血量差为 -104.5，胜率仅为 18.0%，性能明显逊色，反映出其在训练过程中可能存在过拟合或策略更新不充分的问题。

综上，SARSA 和 DQN 均在测试中表现出良好的对战能力，其中 DQN 的胜率优势更为显著，适合用于在高维状态空间中学习对抗策略。而 Q-Learning 的测试性能较差，提示该方法在当前设置下难以有效泛化。整体测试结果验证了训练阶段所学策略在实战场景中的有效性。

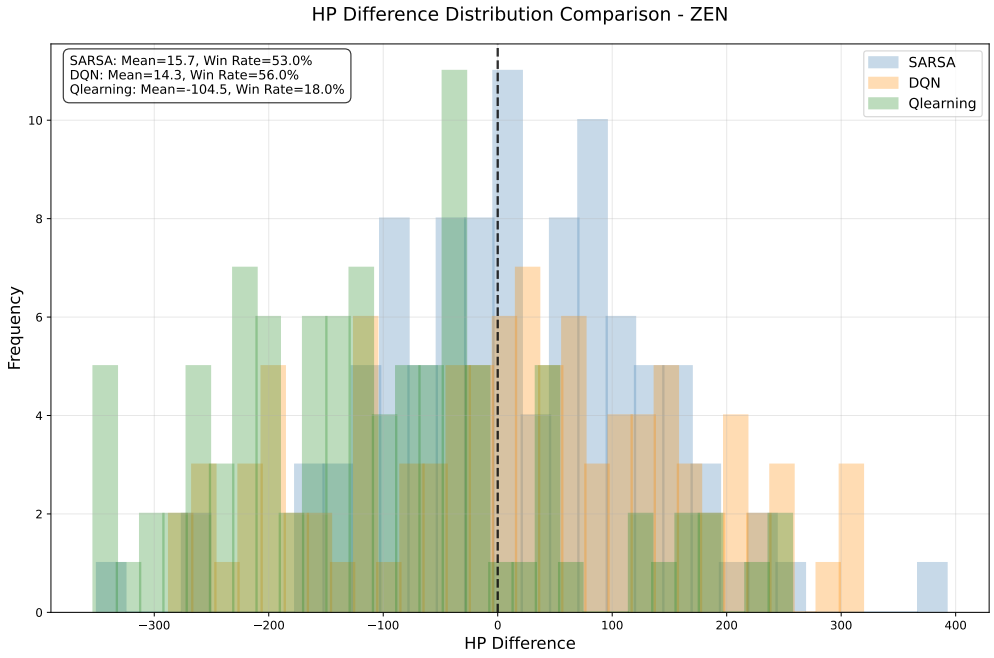

图 4: 三种算法的测试结果统计图

我们进一步在将双方初始血量限制为 200 的条件下进行了测试，以模拟更短回合、更激烈博弈的场景，以 Sarsa 算法为例，100 个测试回合结束时血量差分布如图 5所示。可以看到，Sarsa 在该设定下的血量差分布明显向右侧偏移，即大部分回合都能保持正向血量差，说明其在短局势对抗中表现出色。三种算法的测试结果对比如图 6所示。实验数据显示，在限制血量为 200 的设定下，SARSA 的平均血量差达到 40.1，胜率为 75.0%，在三种算法中表现最为优异，说明其在短局势对抗中能更快做出有效响应。DQN 的平均血量差为 31.6，胜率为 70.0%，整体表现依然稳健，但略逊于 SARSA，可能是由于其策略偏向中长期收益，在局势压缩的条件下反应不如 SARSA 灵活。相比之下，Q-Learning 在该设定下虽较此前有所改善，但平均血量差仍为 -11.4，胜率仅为 44.0%，整体对抗能力仍显不足。

在血量无限和限制情况下的测试结果总结如表2。结果表明，在更高压的对战场景下，SARSA 展现出了更强的即时决策能力和策略适应性，而 DQN 保持了一定的鲁棒性，Q-Learning 则依然难以获得有效优势。

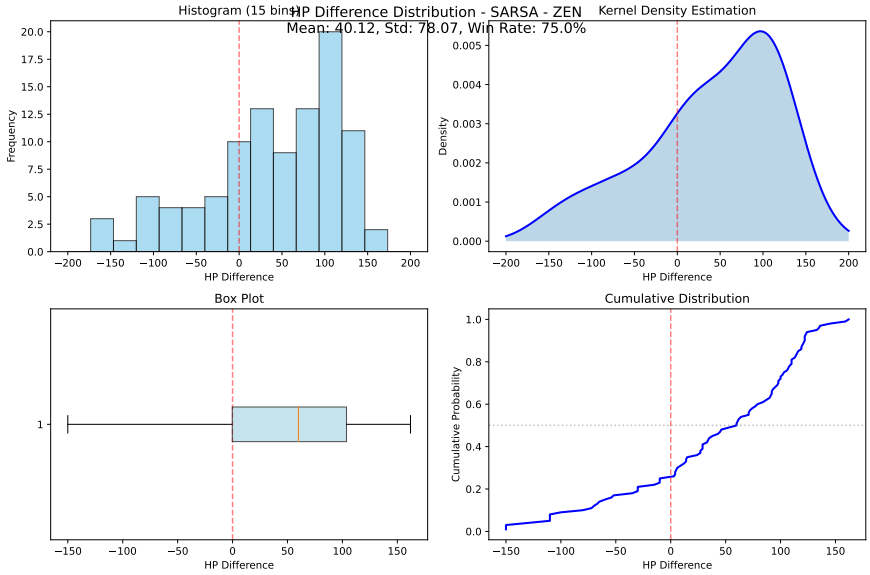

图 5: Sarsa 算法限制血量模式下的血量差分布

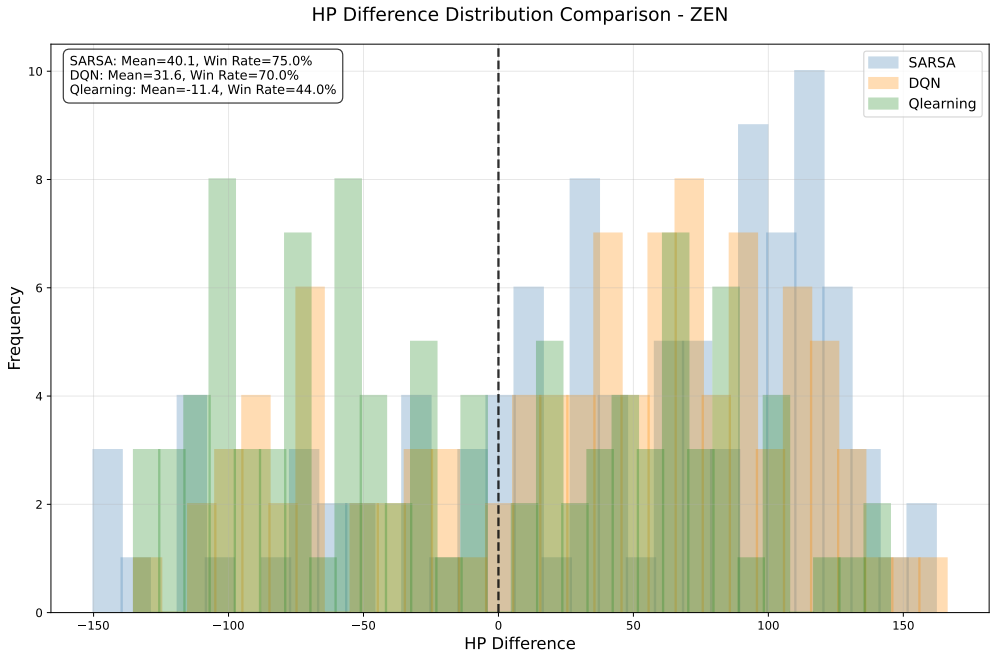

图 6: 三种算法的测试结果统计图（限制血量为 200）

5 结论

本报告围绕基于 FightingICE 平台的格斗游戏 AI 强化学习任务，设计并实现了三种典型的强化学习算法：Sarsa、Q-Learning 和 DQN（Deep Q-Network）。通过与环境的交互，智能体在训练阶段逐步学习对战策略，随后在与固定策略对手 MctsAi 的测试中评估其表现。实验结果表明，Sarsa 和 DQN 均能够较好地掌握有效策略，在标准测试中分别取得了平均血量差为 15.7 和 14.3，胜率为 53.0% 和

表 2: 不同算法在两种测试模式下的性能比较

算法	**标准测试**		**限制血量测试 (HP=200)**	
	平均血量差	**胜率 (%)**	**平均血量差**	**胜率 (%)**
Sarsa	15.7	53.0	40.1	75.0
Q-Learning	-104.5	18.0	-11.4	44.0
DQN	14.3	56.0	31.6	70.0

56.0% 的成绩，显示出良好的对战能力；而 Q-Learning 的平均血量差为 -104.5，胜率仅为 18.0%，性能显著低于其他两者。

在进一步的测试中，我们将双方血量限制为 200，以观察算法在更短对局中的适应性。结果显示，Sarsa 和 DQN 均展现出良好的泛化能力，胜率分别提升至 75.0% 和 70.0%，平均血量差也分别上升至 40.1 和 31.6；Q-Learning 在该设定下亦有所改善，但总体仍处于劣势，胜率仅为 44.0%。

虽然 Sarsa 和 Q-Learning 均采用表格形式存储 Q 值，但两者在训练稳定性和策略学习质量上的差距提示，Q-Learning 可能存在策略更新不稳定、对初始探索过度依赖或对环境反馈过拟合等问题，导致其在本任务中难以收敛到有效策略。而 Sarsa 由于在每一步中使用实际采取的后继动作进行更新（on-policy），在与环境持续交互中更容易获得稳定的估计。

综上所述，DQN 在处理高维状态特征时表现出较强的泛化与学习能力，而 Sarsa 在训练效率与策略稳定性方面同样具有竞争力。Q-Learning 则需要进一步的改进，例如策略更新方式的调整、探索机制的优化或加入平滑更新等技术，以提升其实用性。