南京大学

本 科 毕 业 论 文

院　　系　计算机科学与技术系

专　　业　计算机科学与技术

题　　目　基于值函数的深度强化学习技术在复杂 RTS 游戏场景下的应用与实现

年　　级　2016　学 号　161220124

学生姓名　王明

指导教师　路通　职 称　教授

提交日期　2020 年 5 月 20 日

南京大学本科生毕业论文（设计、作品）中文摘要

题目：基于值函数的深度强化学习技术在复杂RTS游戏场景下的应用与实现

院系：计算机科学与技术性

专业：计算机科学与技术

本科生姓名：王明

指导教师（姓名、职称）：路通、教授

摘要：

随着现实问题的规模越来越大、内容越来越复杂、特征越来越多样，在机器学习的框架下找寻一种通用算法的需求越来越高。基于增强学习的 Deep Q-Learning (DQN)是一种通用性很强的算法，能够直接处理裸数据，如像素级数据,能够克服经验数据的相关性和非平稳分布问题,从而取得比较好的结果。DQN已在传统的Atari游戏上取得了良好效果。本文将DQN算法应用到一个新的游戏场景上，即StarCraft2 (SC2) 这个较富挑战性的RTS场景。目前在SC2上的强化学习方法大多数基于策略梯度方法，较少有基于值函数的方法。本文期望探索基于DQN的方法在复杂游戏场景下的应用潜力。除此之外，SC2中还存在时间序列过长的问题，这给奖赏信用分配(credit assignment)问题带来了很大的挑战。通过这项研究，本文将探索如何有效的设计奖赏去减轻这个问题所带来的影响，并希望这些方法和经验对其他的基于值函数的奖赏信用分配问题的解决带来启发。

关键词：深度强化学习；深度Q网络；奖赏信用分配

南京大学本科生毕业论文（设计、作品）英文摘要

THESIS： Application and Implementation of Deep Reinforcement Learning Technology Based on Value Function in Complex RTS Game

DEPARTMENT：Department of Computer Science and Technology

SPECIALIZATION: Computer Science and Technology

UNDERGRADUATE: Ming Wang

MENTOR: Tong Lu

ABSTRACT：

As the scale of real-world problems becomes larger, the content becomes more complex, and features become more diversity, the need to find a general algorithm within the framework of machine learning is increasing. Deep Q-Learning (DQN) based on reinforcement learning is a very versatile algorithm that can directly process raw data, such as pixel-level data, can overcome the correlation and non-stationary distribution problems of empirical data, and achieve better results. DQN was tested on traditional Atari games. This article applied the DQN algorithm to a new game scene, which is the more challenging RTS scene of StarCraft2 (SC2). At present, most of the reinforcement learning methods on SC2 are based on the policy gradient method, and there are few methods based on the value function. Through this attempt, this article looks forward to exploring the application potential of DQN-based methods in complex game scenarios. In addition, there is also a long time horizon in SC2, which brings great challenges to the problem of reward credit assignment. Through this research, the article will explore how to effectively design rewards to mitigate the impact of this problem, and hope that these methods and experience will bring inspiration to the solution of other value-based reward credit allocation problems.

KEY WORDS: Deep reinforcement learning; Deep Q-Network; Credit assignment

目录

第一章 绪论

1.1 研究背景

近些年随着深度学习的兴起，使得强化学习重新焕发了生机，许多学者致力于结合深度学习和强化学习以达到在某些复杂任务上的不错表现。深度强化学习结合了深度学习超强的将高维信息特征转化为低维信息特征的能力以及强化学习的奖惩反馈机制，在许多规模庞大、特征复杂的领域取得了惊人的效果[4]。Google DeepMind 公司开发出来的 AlphaGo 连续击败了人类顶尖围棋选手，后又更新出 AlphaGo Zero，轻易地击败了 AlphaGo[5]。对于围棋这一种规则复杂、问题空间庞大的竞技游戏，深度强化学习已经展示了其出色的学习能力以及统治级别的表现力。事实上，现在的深度强化学习为自动驾驶、机械控制、文本语言理解、策略游戏、智能问答等领域带来了新的血液。

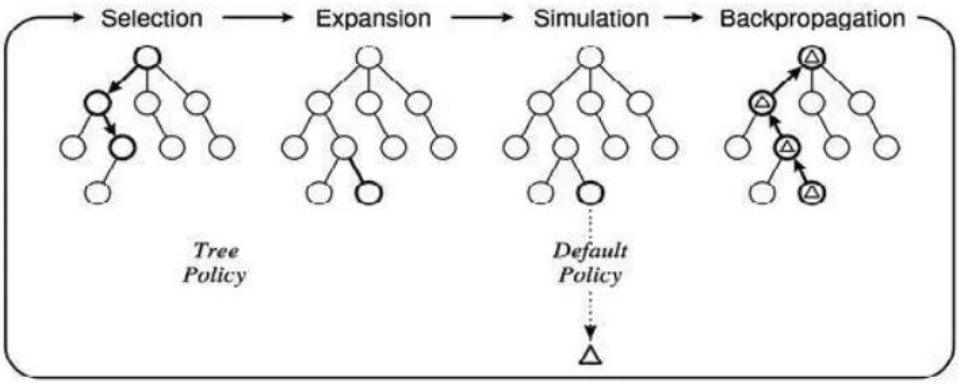

图 1 蒙特卡洛树结构

随着现实问题的规模越来越大，内容越来越复杂，特征越来越多样，在机器学习的框架下找寻一种通用算法的需求越来越大。游戏场景是一个非常适合深度强化学习来施展拳脚的场景。AlphaGo 所适用的围棋其实也是一种游戏，事实上早期的深度强化学习工作也是依赖 Atari 等游戏平台来做测试。游戏的种类丰富，类型多样，规则复杂，问题空间庞大，对于深度强化学习算法来讲是很大的挑战，但是又极大地推动了深度强化学习的发展。在深度强化学习领域，目前的

工作主要分成基于值函数的方法和基于策略梯度的方法。打败人类职业星际选手的 AlphaStar 便是采用了基于策略梯度的方法，如 PPO 等，可以说在该领域已经是目前的最高水平[6]。

本文的重点是放在基于值函数的方法的开山方法——DQN，本文将着重研究 DQN 在复杂 RTS 领域下的应用与实现[4]。本文希望借助基本的 DQN 方法，结合星际争霸 2 这一款特定的游戏，优化奖赏信用分配，探索基于值函数的方法在星际争霸 2 游戏中的表现和潜力。本节将介绍深度强化学习相关的知识以及奖赏信用分配问题。

1.1.1 强化学习

强化学习是机器学习下面的一个分支，对于强化学习的一个智能体来讲，需要把它放到一个特性的环境中去交互，首先需要设定智能体初始的状态，然后在当前的时刻 t 下，在当前的状态 s 下，智能体从可以采取的动作中采取一个动作 a，此时智能体会转移到一个新的状态，同时会获得一个奖励 r，然后时刻 t 变成 t+1。这样的过程一直进行下去，最后的结果可能是完成任务或者任务失败。强化学习与监督学习的区别就是强化学习中没有一个确定的标注好的结果，有的只是每一步得到的奖励，而且无法保证局部最优奖励就是全局最优奖励，因此强化学习的目标就是使得最终得到的累计奖励最多，或者使得最终得到的累计损失最小。所以强化学习本质上是一个试错的过程，通过不断的试错得到的反馈来优化智能体的行为，从而训练出在某种任务上有良好表现的 AI[7]。

强化学习的模式很适合游戏领域，事实也证明其在游戏领域有着强势的表现。

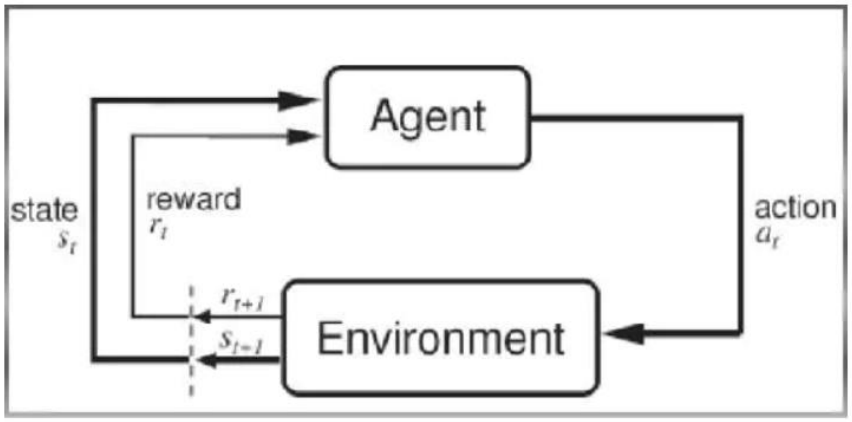

图 2 典型强化学习过程[4]

1.1.2 深度学习

深度学习也是机器学习下面的一个分支[8]。深度学习是相对于浅层学习而言的，早期的机器学习仅仅采用简单的特征和对应的标签来学习的模式，一般对于较为复杂的问题，只能人工手动选取和抽象特征作为输入，使用浅层的神经网络来进行特征和标签的学习[18]。这样做的坏处是无法处理原始的高维数据，并且人工的特征提取会带来误差。深度学习便可以解决浅层学习的问题，深度学习是指具有较多的隐藏层的神经网络，深度神经网络中间包含隐藏层，每个隐藏层可以对上一层的输入进行一个非线性变换来抽象成更高层的特征然后输出到下一层，这样整个学习过程不需要人工的筛选，自动学习，可以得到更加精确的结果。

深度学习被广泛应用在人脸识别、自动驾驶、医疗图像等领域，深度神经网络对于海量数据训练的能力很强，理论上对于任意的输入和任意的输出都可以拟合。由于硬件的发展，现在深度学习的应用范围更广泛了，对于很多领域来讲，深度学习都可以作为一个非常实用的工具来拟合复杂的函数[19]。

1.1.3 深度强化学习

深度强化学习结合了深度学习和强化学习的优点，使用卷积神经网络将高维特征空间转变到低维特征空间，然后再利用强化学习的奖赏信用分配机制来进行

决策优化，改善智能体的行为。深度强化学习是端到端的，直接从原始的高维输入到决策的过程。

深度学习的代表是 DQN，将卷积神经网络和 Q-Learning 相结合，就可以接受像素级别的数据输入，训练出的模型在很多游戏中发挥了超过人类的水平[17]。DQN 是基于值函数的深度强化学习方法，还有一个大类是基于策略梯度的方法。著名的 AlphaGo 就是使用卷积神经网络和策略梯度相结合的办法，不断进行自我对弈，从而提升围棋对决水平[9]。总的来说，深度强化学习学习能力强大，具有处理复杂问题的能力，除了在游戏领域有杰出成就之外，将来必然会有更加广泛的应用。

1.1.4 奖赏信用分配问题

对于 DQN 而言，奖赏信用分配问题是一个很重要的问题，对于智能体性能的提升有着很关键的作用。对于智能体采取某一个动作 a 后，得到了奖励 r。那么假设你经过一些列的动作最后完成了任务，其中得到了 r1,r2,r3……rn 等一系列的奖赏，这个时候你需要知道这之前的一系列动作当中，哪个动作对于现在顺利完成的结果贡献最大，或者说这一系列动作每个动作对于最终结果的贡献比例是多少，这个问题就是奖赏信用分配问题。这个问题影响了你将如何优化你的每一步决策，也会影响训练模型的效率。目前比较流行的方式是使用马尔科夫决策模型，每次动作的决策只与智能体当前的状态和当前能执行的动作有关。

1.2 国内外研究现状

1.2.1 深度强化学习研究进展

自从 DQN 被提以来，其在 Atari 等游戏上的高水平表现激发了深度强化学习领域研究的热情。DQN 是首个将卷积神经网路（CNN）与强化学习中的 Q-Learning 相结合的算法，通过检验回放和目标网络技术来克服经验数据的相关性和非平稳分布的问题。后来各路研究人员在 DQN 的基础上又提出了 Double DQN、Dueling

DQN、Averaged DQN、Rainbow DQN 等新的算法来解决原始 DQN 学习抖动性过大、学习性能低等问题。但是由于 DQN 无法处理更加复杂的场景，因此在相对复杂的游戏环境下，比如星际争霸 2，相对于基于值函数方法的 DQN，研究者更加倾向于基于策略梯度的方法。近年来，研究人员已经研究出 DDPG、TRPO、PPO 等算法。基于以上算法训练的智能体，已经可以在各路游戏中击败非常厉害的人类玩家。目前更加前沿的研究可能更加倾向于从单个智能体的训练发展为多个智能体的训练，这涉及到智能体之间的交互，需要处理的问题必将更加复杂，但同时深度强化学习所能适用的领域也将更加多元[11]。

1.2.2 奖励机制的研究

奖励机制的研究对于 DQN 来讲是一个重要的话题，目前用的较多解决方法是使用马尔科夫决策模型。该模型假设每次动作的决策只与智能体当前的状态和当前能执行的动作有关，考虑到环境的复杂性以及决策的多样性，同样的动作可能得到的回报并不一定一样。越是往后走，越不确定能获得多大的奖励。因此，本文希望弱化未来奖励的贡献比例，通常的做法是对于未来的奖励我们要乘上一个 0-1 之间的折扣因子，这样就可以调整未来的奖励的贡献度。这样做确实是比较合理的做法。

由于奖赏机制在强化学习中起着反馈信号的作用，可以说对学习有着非常大的影响。目前也有不少研究人眼试图设计一些特殊的奖赏机制来提高学习效率。比如逆强化学习，是利用已知的最优的决策来反过来学习奖赏机制的设计。还有比如混合奖赏学习，一个任务被分成许多阶段，每个阶段使用不同的奖赏函数[10]。

1.3 研究内容

基于强化学习的 Deep Q-Learning(DQN)就是一种通用性很强的算法，能够直接处理裸数据，如像素级数据，能够克服经验数据的相关性和非平稳分布所带来的问题，从而取得相对较好的结果。DQN 一开始被提出时是在传统的 Atari 2600

游戏上做了测试。在本篇文章中，本文将 DQN 算法应用到一个新的游戏场景上，应用到星际争霸 2 这个较富挑战性的 RTS 场景。目前在星际争霸 2 上的强化学习算法大多数使用基于策略梯度方法，较少有基于值函数的方法。

通过这次尝试，本文期望探索基于 DQN 的方法在复杂游戏场景下的应用潜力。除此之外，SC2 中还存在了时间序列过长的问题，这给奖赏信用分配(credit assignment)问题带来了很大的挑战。通过这项研究，本文将探索如何有效的设计奖赏去减轻这个问题所带来的影响，并希望这些方法和经验对其他的基于值函数的奖赏信用分配问题的解决带来启发。

1.4 论文组织结构

本文之后的章节的主要内容如下所述：

第一章：本章简要介绍本研究的相关研究背景，简要说明本文的研究内容和意义，陈述本文的组织结构。

第二章：本章简要叙述当前深度强化学习的主流建模方法、网络架构方法和两种主流算法。

第三章：本章简要介绍在游戏领域应用深度强化学习的前人的研究成果，论述本文从中得到的启发以及提出相应的研究内容。

第四章：本章详细论述实验设计流程并展示和分析实验结果，得出实验的结论。

第五章：总结本文的主要工作，并提出未来可能的在该问题上的后续研究方向。

第二章 相关工作

深度学习赋予了神经网络极强的学习能力，深度神经网络、卷积神经网络、循环神经网络等特定架构的深度神经网络又赋予了神经网络在不同应用场景下的良好的表现。强化学习秉承智能体自己和环境交互，自我学习的理念，极大提升了自学习的能力。深度强化学习则结合了两者的优点，以奖励机制作为导向，在许多任务上的表现已经超越人类。本章将介绍游戏领域，深度强化学习的相关工作和经典算法。

2.1 DQN 在 Atari 游戏中的应用

2.1.1 主要工作

在 2013 年，DeepMind 团队首次将深度学习和强化学习相结合，提出了第一个深度强化学习算法 Deep Q-Network，这也是深度强化学习领域的开山之作[4]。DeepMind 团队提出了端到端的算法模型，其采用卷积神经网络，可直接从高维感官输入中去学习游戏的控制策略，训练方法是 Q-Learning 的变体，最终可以直接接受原始像素级别的输入，输出估计未来奖励回报的价值函数。研究者在七个 Atari 2600 游戏上做了实验，发现模型在 6 个游戏中的表现比之前的工作要优秀，并且在其中 3 个游戏中可以获得超越人类的表现。

2.1.2 Deep Q-Network

DeepMind 的研究者首先对于 Atari 问题进行了建模，将问题变成了计算机可以理解的决策过程。整个决策任务的流程是：智能体处于起始状态 S0，采取了一个动作 A0 后转移到了状态 S1，并且得到了一个相应的奖励 R0，然后又采取了一个动作 A1 后转移到了状态 S2，相应得到了一个奖励 R1。按照这样的形式不断地进行下去直到智能体到达一个终止态，终止态可以是胜利或者是失败，在此

状态下智能体不会有任何的动作，此回合的任务就结束了。结合到具体的 Atari 游戏，状态集合 S 就是游戏界面的集合，也就是每一帧游戏界面的截图的集合，这个数据量非常的大，DeepMind 使用了卷积神经网络来大大减小了数据量（包括界面的分辨率），同时又最大程度保持数据分布一致性。动作集合 A 采用了游戏本身可以进行的动作，如人物移动和攻击防御指令等等。奖励 R 直接采用游戏的评分机制或者根据最终的输赢进行 0-1 评分。

模型建好之后，DeepMind 团队提出了如下图所示的经典的 DQN 的算法[21]。

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

图 3 DQN 算法流程[21]

该算法将 Q-Learning 的强化学习算法进行了一个优化，使得更新 Q 表的操作变成了更新神经网络参数，利用 Q-Learning 的学习算法，结合随机梯度下降，不断优化神经网络参数，使得决策向未来奖励回报最大的方向前进。

2.1.3 经验回放和目标网络

DeepMind 研究者发现，与监督学习发现输入数据和标签数据之间直接的关联相比，游戏场景中一个动作和因它得到的奖励之间的延迟可能会非常长，以及游戏场景中经常会遇到高度相关的一个状态动作序列，这些数据间的相关性会很大程度影响学习的效率。为了克服经验数据相关性，经过研究，DeepMind 提出了 DQN 的两大利器。它们分别是经验回放和目标网络[12]。经验回放是指 DQN

算法可以做到离线学习，也就是说并不需要对当前的步骤下进行学习，每当获得一个转移样本（转移样本的形式为 S，A，R，S_，其中 S 代表当前的状态，A 代表当前状态 S 下采取的动作，R 代表该动作获得的奖励，S_代表在 S 下采取了动作得到的新的状态）时，可以将该转移样本存储起来，当样本的数据多了以后，随机从中抽取样本进行学习，这样就很好地克服了前后样本之间相关性较大的问题，减少了相关性对学习效果的影响，同时也使得同一个转移样本可以被学习多次，大大提高了学习的效率[20]。目标网络是另一个克服数据相关性和提升学习稳定性的利器。DQN 算法建立两个架构完全一样的网络分别为 evaluation-network 和 target-network，其中 evaluation-network 负责具体的训练，参数更新，但是在计算 Q 值的估计值时使用 target-network 的参数，经过一定的时长后将 target-network 的参数全部替换为 evaluation-network 的参数。这样做的好处是保持 Q 估计的稳定性，提升了学习的速度。

2.1.4 DQN 的改进

虽然 DQN 非常强大，并且可以完成从原始图像输入到直接决策输入的端到端的处理架构，但是仍然存在学习的抖动性较大、状态过多产生 Q 表爆炸、不能用于连续动作等一系列问题。后来的研究者们提出了 Double DQN、Dueling DQN、Averaged DQN 等一系列改进的算法从 DQN 参数的更新方式、DQN 网络架构、target-network 参数替换等方面对原始的 DQN 进行了优化[1][2][3]。

2.2 深度强化学习在 StarCraft2 中的应用

2.2.1 主要工作

StarCraft2 是一个大型的对战类游戏，因为它的状态空间庞大、对局时间长，而给深度强化学习的研究者带来了很大的挑战性[13]。科研人员为了解决在真实环境中训练时间过长的问题，提出了引入思想游戏的概念，首先在思想游戏中对于智能体进行训练，得到一个相对高水平的模型，然后再放入真实的环境中

进行第二阶段的训练。训练方法使用了基于策略梯度的 Proximal Policy Optimization 算法[14][15]。最终在地图 Simple64 中以百分百胜率打败了内置的第七级难度的机器人，训练的速度相较于之前可提升 100 倍。

2.2.2 思想游戏

研究人员经过探讨，认为 StarCraft2 场景具有挑战性，除了因为它庞大的状态空间、庞大的动作空间、庞大的游戏空间（地图大、视野大）、以及不完善的信息（如对手的情况在没有探测的情况下是不清晰的），根本是因为游戏的规则依赖性太强，比如你需要发动进攻指令，就必须先生产士兵、而生产士兵就必须先生产兵工厂、生产兵工厂就必须先收集资源。因此长长的依赖链大大提升了训练的复杂度。研究人员巧妙的提出了一个思想游戏，该思想游戏是对于真实 StarCraft2 游戏的一个简化的模拟。该模拟去除了很多 StarCraft2 中复杂的元素，留下来一些对于胜负而言非常关键的元素（比如建筑物和士兵信息）。这样每次在思想游戏中的某一个状态进行决策，将动作输入到真实的游戏环境中得到所对应的奖励和下一个状态，然后将这个新状态映射到思想游戏中，得到思想游戏中的新状态。所以这种方式的巧妙之处在于，极大减少游戏复杂度和计算量的同时又可以高度模拟真实游戏场景。有了思想游戏之后，就可以先在思想游戏中进行预训练，得到一个性能不错的智能体，然后在放入真实游戏环境进行训练，此时训练需要的时间就大大缩小了。

2.2.3 难度控制和训练算法

在思想游戏这个模型中，研究人员主要使用了 ACRL 和 PPO 两个算法。

ACRL 算法用来对游戏难度进行控制，StarCraft2 内置了不同难度的机器人，ACRL 通过设定一个胜率阈值，计算和监督智能体在当前难度下的胜率情况，当智能体在某一个难度上的胜率表现超越了这个阈值后，就可以判定在该难度上的训练已经达到效果，就可以加大难度进行下一阶段的训练。

PPO 是一种基于策略梯度的方法。它的一个很大的优势是可以解决动作空间连续的问题，因为基于策略梯度的方法是用概率分布来表达每个动作选取的可能性[16]。这种方法使用一个参数向量去表达每一个策略，是一个关于状态集合 S 和动作集合 A 的函数，智能体使用某种决策来执行一段轨迹的时候，可以得到一个整体的期望奖励，而此方法的目标就是优化这个参数向量，使得其获得的期望奖励最大。基于策略梯度的方法借鉴了 DQN 的经验回放和目标网络的方法，将每一次产生的转移样本都存储到一个记忆库中，每次学习的时候随机去抽取一个小的样本来进行学习，打乱了数据之间的相关性。同时 PPO 也同样建立了目标网络，定期用当前网络的参数去替换掉目标网络的参数，但是并不是像 DQN 那样完全一样的复制，而是采取了一种小幅参数更新的方式，这样会使得学习更加稳定。基于值函数的方法主要按照下图的方式进行更新，对于当前状态 St 对应的每个动作 Am 的价值计算，是利用执行完 Am 动作之后得到的状态的 St+1 的价值与得到的奖励 Rt 来进行修正的。这样就形成了一种类似自助法学习的过程，非常得快速且高效。

2.3 本章小结

本章介绍了两类有关将深度强化学习技术应用在游戏场景下的相关工作和经典算法。一类是 DeepMind 团队提出来的端到端的 DQN 算法，在 Atari 2600 游戏下取的了惊人的效果。另一类是研究者在 StarCraft2 上应用了基于策略梯度的 PPO 算法，并且提出了思想游戏的概念来加快训练的速度，最终也取的了比较好的效果。同时，本章还阐述了两种不同类型的方法的异同和分别的改进算法。

第三章 基于值函数的 DRL 在 SC2 场景下的应用

本章将介绍本次研究的研究目标，叙述 StarCraft2 游戏场景的建模以及 DQN 的运作流程，并且探讨关于奖励机制的改进的方法。

3.1 研究目标

本文试图将经典的深度强化学习算法 Deep Q-Network 应用在 StarCraft2 这一复杂的 RTS 游戏场景下，期望训练出一个可以击败 StarCraft2 内置的七级难度机器人的智能代理。为了取得更好的实验效果，本文还将对 StarCraft2 奖励机制进行探索，尝试通过改进对于训练效果影响很大的奖励来近一步提升训练效果，期待为后人在复杂游戏场景下的研究提供一定的启发。

3.2 StarCraft2 游戏场景

最初的深度强化学习算法以及后续的研究很多都是在 Atari 2600 游戏环境中做的实验和研究。Atari 游戏场景的特点是：2D 的有图形界面的游戏、分辨率不高能看出明显的像素锯齿、玩家的可选动作比较有限、游戏所要达成的任务相对来说没有那么复杂。虽然在 Atari 上的研究取得了非常好的成果，但是 Atari 的场景还是过于简单，为了提升算法的泛化能力，本文选取 StarCraft2 这一复杂的 RTS 游戏作为研究目标。

StarCraft2 是暴雪游戏公司在 2010 年推出的一款即时战略性游戏，它的前身是 StarCraft1，与大多数即时战略性游戏一样，目标都是通过双方不断的对战而击败对手。作为玩家，你可以对自己这一方面的军队、建筑进行全局的指挥和控制，争夺战场资源的控制权，最终打败所有敌人。游戏开始之前，玩家需要选择自己的种族，种族的种类有人类、星灵和异虫。每个种族都有不同的建筑物和士兵，也都拥有不同的作用和进攻手段。最基础的兵种是工人，他们会采集地图上的资源，从而能建造更多的建筑以及庞大的军队。一些高级的建筑除了需要

满足建造资源外，还需要你的基地达到特定的要求，比如需要建造特定的建筑、研发相应的科技。走上胜利的常规之路是合理建造一支兵种合理、作战能力强大的军队而一举摧毁对手。这款游戏的复杂的地方还包括兵种的特殊技能、对手信息不明确。兵种的特殊技能是大规模作战时必须考虑的事情，如何调整阵型使得前排能够抗住火力、后排能够放心输出、医疗队能够及时救援，如何针对对手核心，优先摧毁对手杀伤力大的兵种，如何采用包围战术、声东击西战术、操控多支军队执行联合作战，是高级玩家和普通玩家拉开差距的地方。对手的信息一开始是不明确的，战场始终存在一种战争迷雾的东西，你无法不通过侦查而了解对手的建筑和军队状况，这样带来的弊端是你无法有针对性的建造士兵、以及你不知道对手的会采取什么样的战术，所以侦查与反侦察也是 StarCraft2 里比较重要的手段。

图 4 星际争霸 2 对战图

StarCraft2 状态空间和动作空间庞大、规则依赖性强、对手信息不明确，因此是很大的挑战。本文将在前人工作的基础上，充分利用前人的研究去解决非本次研究关注的问题，从而更加专注于实现和改进 Deep Q-Network 这一核心工作。

3.3 StarCraft2 问题建模

强化学习与监督学习有一个很大的区别就是强化学习每一步决策都无法直接获得一个确定性的标签信息，无法通过确定性的标签信息来进行相对精确的学习。那么在强化学习中，往往是用一个奖赏来代替监督学习中的标签，来指导智能体的行为。对于强化学习适用的这一大类问题，主流的方法是采用马尔科夫决策模型。对于 StarCraft2 场景，本文同样使用马尔科夫决策模型。该模型可以使用一个六元组来表达（S，D，A，P，r，R）：

S：所有状态的集合，也就是智能体在环境中可以存在的所有的状态的集合。对于 StarCraft2 场景来说，如果像 Atari 游戏那样，以游戏原始像素数据作为状态，那将是不可想象的庞大。因此本文选择沿用了前人研究的思想游戏的框架，简化状态信息，将游戏看成一个棋盘，将建筑物、士兵种类和数量等关键性信息视作状态。

D：这是一个关于 S 的概率分布，表达的是初始情况下，智能体处在各种状态的概率。对于 StarCraft2 场景来说，当选定了 Simple64 这一特定的地图之后，初始的状态都是固定的，只会根据不同的种族（人族、神族、虫族）会略有不同。

A：所有动作的集合，也就是智能体在环境中各种状态下所能采取的所有的动作的集合。对于 StarCraft2 场景来说，就是真实人类玩家可以执行的操作，比如收集资源、建造工厂、生成士兵、进攻指令等等的集合。

P：状态转移函数，是一个关于 S 和 A 的概率分布，表达的是智能体在状态 S 时，采取了动作 A，转移到其他状态的概率分布。对于 StarCraft2 场景来说，也就是对于某个状态(当前的建筑物、士兵、受攻击状况)等情况下，根据决策采取了一个动作，获得了一定的奖励，同时进入了一个新的状态(如建筑物和士兵的数量变化了)。

r：衰减因子，对于未来奖赏的一个衰减，越是往后的步数，得到的奖赏的贡献比例就会越小，呈指数级下降。表示更加看中当前步骤带来的奖赏而对于未来奖赏看的低一些。

R：这是一个关于 S 和 A 的函数，记录了智能体处于状态 S 时采取了动作 A 之后所得到的具体的奖赏。对于 StarCraft2 场景来说，可以有多种方式计算奖励，可以是最终的胜利或失败，也可以是当前的建筑物、兵力状态的加权值。

基于马尔科夫决策模型，整个决策任务的流程是：智能体处于起始状态 S0，采取了一个动作 A0 后转移到了状态 S1，并且得到了一个相应的奖励 R0，然后又采取了一个动作 A1 后转移到了状态 S2，相应得到了一个奖励 R1。按照这样的形式不断地进行下去直到智能体到达一个终止态，终止态即是 StarCraft2 游戏最终判胜或者判负，在此状态下智能体不会有任何的动作，此回合的任务就结束了。

这样一个过程中，可以计算出该回合中累积的折扣奖励:

$$\text{Rewards} = R0 + r * R1 + r^2 * R2 + r^3 * R3 + \cdots\cdots + r^k * Rk$$

这个值就是该回合任务中智能体得到的奖励，本文的目标就是要最大化这个期望回报[4][21]。如果达到了这个目标，那么就得到了针对该任务的比较好的决策。

3.4 Deep Q-Network 在 SC2 下的应用

首先本文对于 StarCraft2 问题进行了马尔科夫决策模型的建模，在此模型下，得到的是形如(S_t, a, r_t, S_{t+1})的转移样本，接下来本文使用 DQN 来对这些转移样本进行学习。DQN 是基于值函数的方法的典型代表，它具有离线学习的能力。DQN 结合了卷积神经网络和 Q-Learning 学习方法。卷积神经网络非常擅长处理图像等，对于简单的游戏场景而言，可以直接将界面截图作为神经网络的输入，通过卷积神经网络之后大大降低了输入数据的特征维数，使得学习的计算量控制在一个可接受的范围呢。Q-Learning 自助学习方法可以实现对某个状态 Q 值的单步更新，其更新的方式如下图所示。Q 表中的值表示的就是 Q 值的估计值，对于 StarCraft2 来说就是 Q 表中当前状态S_t下选取动作a得到的 Q 值。Q 值的现实值可以用下一个状态的 Q 值估计最大值乘上一个衰减因子加上获得的奖励来表达，对于 StarCraft2 而言就是获取状态S_{t+1}下 Q 的最大值，乘上一个衰减因子γ，再加上从S_t到S_{t+1}的奖励r_t，每次学习的就是 Q 值的估计值和现实值的差值。Q-Learning 更新策略是智能体具有学习能力的核心，它表达的是通过奖励来不断最大化每个动作的累计回报，最终使得整个回合或者流程的累计回报最大[4][21]。

$$V^{\pi}(S_t) \leftarrow V^{\pi}(S_t) + \alpha(r_t + \gamma V^{\pi}(S_{t+1}) - V^{\pi}(S_t))$$

Q-Learning 的更新是在 Q 表中进行，而 StarCraft2 的状态和动作空间过大，无法用 Q 表来表达。DQN 在 Q-Learning 的基础上添加了深度神经网络，接收的输入为状态，输出每个动作对应的 Q 值，将对于 Q 表的更新策略变成了更新神经网络参数，每次神经网络输出的 Q 值视作估计值，真实值按照下图所示的方式计算，用 Q 值估计最大值乘上一个衰减因子加上获得的奖励来计算，然后神经网络就用来学习真实值和估计值的差[21]。对于 StarCraft2 而言，输入为当前的状态S_t，输出为在当前状态下采取每个动作获得的 Q 值，然后将状态S_{t+1}作为神经网络的输入，在输出的 Q 值中选取最大的 Q 值，然后乘上衰减因子γ，加上从S_t到S_{t+1}的奖励r_t，就得到了状态S_t的 Q 的现实值。

$$Y_{qlearn} = r_t + \gamma maxQ^{\pi}(S_{t+1}, a)$$

就像前文提到的那样，DQN 的两大利器经验回放和目标网络使得 DQN 变得强大。具体而言，对于每局 StarCraft2 的游戏对战，本文都会将对战的每一步按照转移样本的形式记录在存储池中，这样每次都从中随机抽取部分样本进行学习，这样不仅可以克服数据间的相关性，而且每个样本有机会被重复利用。同样本文会建立两个神经网络，一个是评估网络 q_eval，一个是目标网络 q_target，其中评估网络负责具体的训练，参数更新，但是在计算 Q 值的现实值时使用目标网络的参数，经过一定的时长后将目标网络的参数全部替换为评估网络的参数。这样做的好处是保持 Q 估计的稳定性，提升了学习的速度。

3.5 奖励机制的探究

强化学习中，奖励的研究也是很重要的一个领域，如何有效利用奖赏以及如何对对应的应用场景设计奖赏也是一个非常值得探究的领域。DQN 作为的经典的强化学习算法，其对于奖励的处理其实是非常合理的，尤其因为定义了衰减因子，使得未来回报的比重降低，更加看重当下的回报，符合正常的游戏思路。另外如果对于 DQN 中的奖励计算进行修改很可能出现训练速度过慢或者无法收敛的问题。因此本文希望能从游戏环境的角度入手，结合 StarCraft2 这一具体的场景，设计不同的游戏奖惩规则，尝试在该方面做一些探索。本文将首先使用具体的依靠胜负来奖励的 0-1 奖赏模式，其次本文希望尝试去基于实时环境信息的，通过

一定的加权来计算当前动作的奖励。

3.6 本章小结

本章首先阐述了本次研究的研究目标，其次介绍了 StarCraft2 这一款即时战略性游戏的游戏模式、对战内容，然后采用马尔科夫决策模型对于此游戏场景进行了建模，最后详细叙述了适用于此游戏场景的具体的基于 DQN 的算法模型。除此之外，对于对实验效果影响比较大的奖励机制的设计进行了探索，分析了 DQN 对奖励的使用方式并提出了改进方法。

第四章 实验设计和结果分析

本章首先简要介绍了本次实验的整个流程，其次说明了本次实验开展的硬件和软件环境，之后展示了多组实验的结果并对实验结果进行了分析，最后对于本次论文的实验部分做一个总结。

4.1 实验流程

本文的目标是探究基于值函数的深度强化学习技术在复杂RTS游戏场景中应用与实现，其中深度强化学习技术选取的是非常具有代表性的Deep Q-Network，游戏场景选取的是 StarCraft2。由于 DQN 算法本身比较复杂、涉及到的参数很多，以及 StarCraft2 同样是一个状态空间和动作空间很大的复杂游戏，因此本文考虑先在OpenAI 小游戏环境中实现 DQN 算法，这里选取的小游戏是 CartPole 倒立摆，观察算法的效果，从而对最终的在 StarCraft2 中使用 DQN 算法提供基础。

所以本文先在 CartPole 倒立摆游戏场景中实现 DQN 算法，比较不同的参数下的 DQN 的实验效果。然后将 DQN 适配到 StarCraft2 游戏环境中，从 DQN 参数调整的结果、不同游戏难度下的 DQN 的结果、与 PPO 算法的比较的结果、不同奖惩机制下 DQN 的结果等四个方面来全面评估 DQN 在 StarCraft2 中的表现。最后得出本次实验的结论。

4.2 实验环境

实验硬件平台：由于 StarCraft2 较为复杂，对局一场游戏需要较长的时间，在普通计算机用 CPU 跑不现实，因此使用了实验室的一台 Linux 服务器，服务器的具体配置如下所示：

表 1 服务器配置

设备	配置
CPU	Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz 1 颗 40 核
显卡	NVIDIA Tesla V100-SXM2 32GB
内存	512GB
硬盘	2TB

实验软件平台：本次实验代码全部采用了 Python 编写，主要用到了 StarCraft2 的 Linux 版本的 API 来搭建实验环境，用 tensorflow 来实现了深度神经网络。在实验运行方面，采用了 GPU 来跑，并使用了多线程技术。

4.3 CartPole 实验结果与分析

4.3.1 实验说明

本文首先在 CartPole 倒立摆游戏中做一个初步的实验，目的是正确实现 DQN 算法，观察 DQN 算法的学习效果，并尝试通过调试各种参数的来提升 DQN 的学习效果，同时也初步认识到 DQN 中某些参数的大小对于训练的影响，从而为之后将 DQN 用于 StarCraft2 提供一个良好、完备的基础。

对于 CartPole 小游戏而言，就是一个倒立的棍子放置在一块砖头上，每次能采取的动作就是砖头左移一点点或者右移一点点，然后棍子会根据自己和砖头的相对位置以及自己本身的状态决定自己往左偏还是往右偏，当棍子左偏或者右偏超过一定的角度大小则判定游戏结束，游戏的目标是使得棍子一直保持不倒。因此智能体应该通过 DQN 的训练慢慢学会去保持棍子的直立，游戏的效果评判就是坚持的时间的长短。因此接下来的实验用来评判学习效果的指标就是平均坚持的时长。比如进行 N 局游戏，将 N 局游戏的时长求一个平均作为这 N 局游戏的得分。这里每种算法进行 60 局游戏，比较他们的平均坚持的时长。

4.3.2 DQN 在 CartPole 中的表现

本文设计的对照实验是：一种算法是每一步都随机选择动作，另一种算法是使用 DQN 并且选择相对来说较为合适的参数来决定每一步游戏采取的动作。下图展现了两种不同的算法用在 CartPole 游戏中的效果。

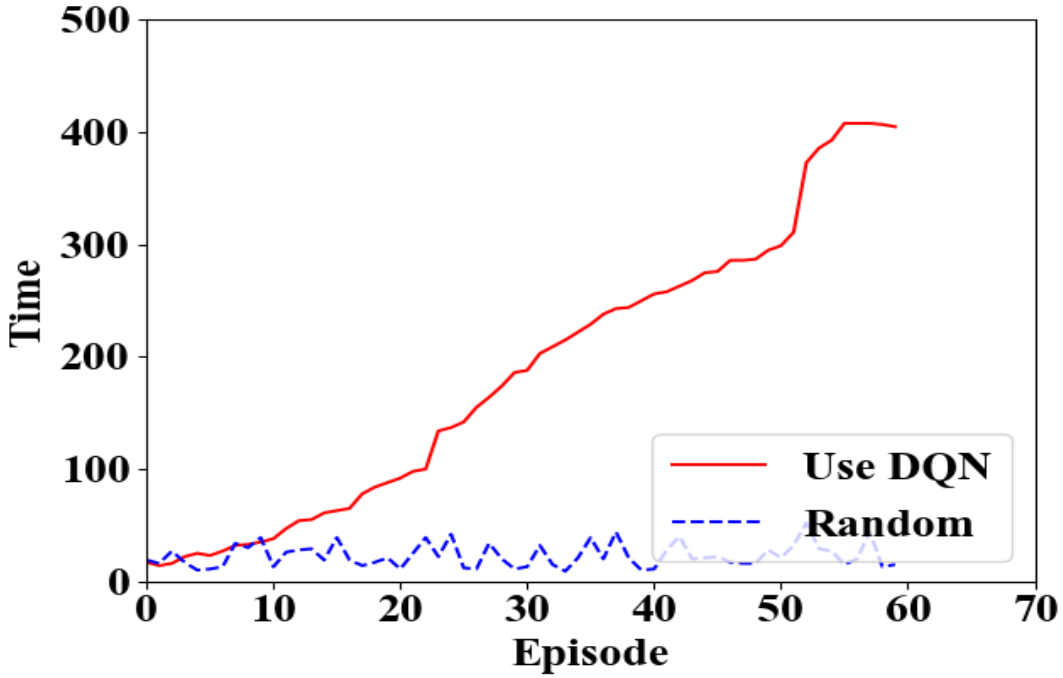

图 5 不同算法在 CartPole 的效果对比

通过实验结果可以发现，如果是完全随机，那么倒立摆能坚持的平均时长大概在 20 到 30 秒之间。而应用了 DQN 算法进行学习之后，可以看到能坚持的时长是在稳定上升的，在第 60 局游戏的时候已经可以做到坚持 400 秒，这充分说明了 DQN 在状态空间和动作空间都很小的任务上是很有效的。

4.4 StarCraft2 实验结果与分析

4.4.1 实验说明

在此实验中，本文将 DQN 算法结合 StarCraft2 的 API 搭建了深度强化学习训练平台。所有的实验数据均在 StarCraft2 的 Simple64 的地图上采用星灵对战

人类，其中星灵智能机器人是使用 DQN 算法来训练和控制的，人类智能机器人是采用 StarCraft2 内置的电脑玩家。本次实验使用了实验室的 GPU 服务器，每组实验使用 10 个线程来跑，每组实验运行 30 次迭代，每次迭代进行 50 场对局，实验的效果图的横轴代表第几次迭代，纵轴代表每一个迭代的 50 场对局的胜率。

4.4.2 不同参数下 DQN 的表现

在此实验中本文将探究随机因子 epsilon、学习率α、衰减因子γ、样本存储池大小 size 对于 DQN 学习效果的影响。

在做决策选取下一步动作的时候，有一个参数 epsilon 用来决定每次动作是采取随机动作，还是经过 DQN 的神经网络选取一个动作。每次生成一个 0 到 1 之间的随机数，如果随机数小于 epsilon，那么将通过 DQN 网络选取动作，反之，采取随机动作。由于 DQN 学习可能会出现很大的抖动，因此加入部分的随机性有时可以获取到更好的效果，也解决了完全由 DQN 选择带来的无正样本的问题。下面是实验设置和实验结果：

表 2 不同 epsilon 的实验设置

编号	学习率	衰减因子	存储池	随机因子
1	0.01	0.9	20000	fixed 0.6
2	0.01	0.9	20000	fixed 0.9
3	0.01	0.9	20000	changed 0.6
4	0.01	0.9	20000	changed 0.9

其中 fixed 代表固定的 epsilon 值，而 changed 代表 epsilon 值从 0 每次累加 0.05 一直到设定的值之后保持不变。

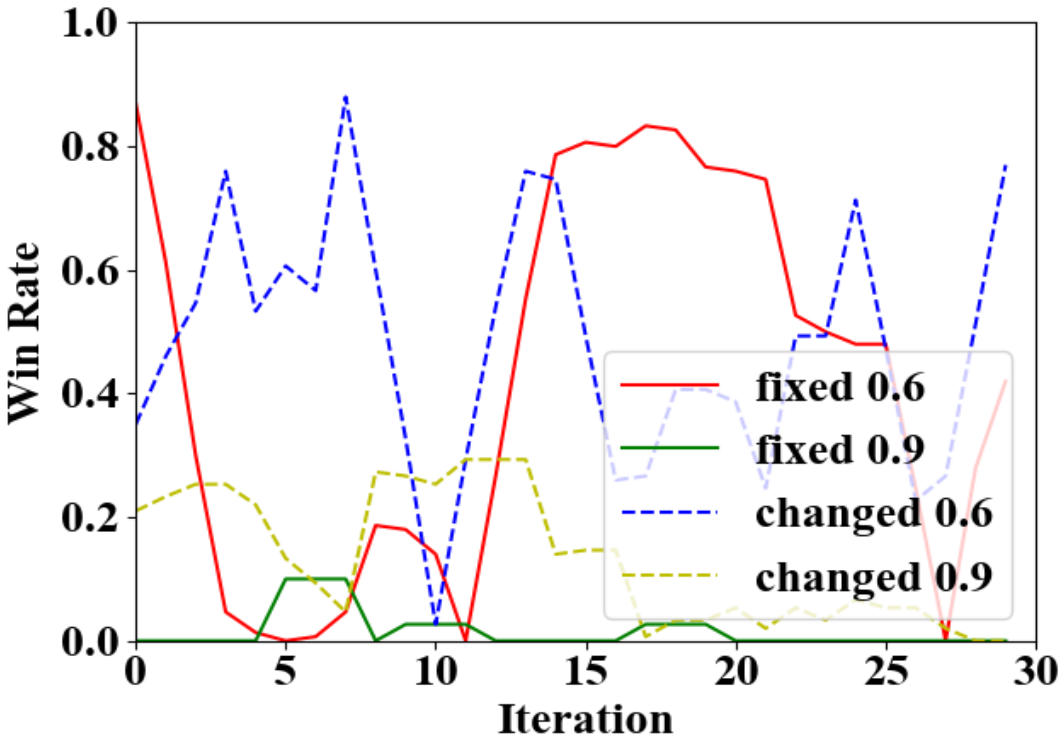

图 6 不同随机因子的效果对比

可以看出 epsilon 的影响还是很大的，整体上使用 changed 模式的效果要好，因为在前期的对局当中，随机的 epsilon 更容易产生正样本。而采用 fixed 模式很有可能在前期训练不充分的时候，保持一个低胜率，从而影响后面的训练。

DQN 每次参数的更新，是学习对当前状态 s 的估计值和现实值的差，而学习率α决定了这次差有多少被学习了。如果学习率过大，那么学习的抖动性将非常大，因此每一个样本的学习，都会对神经网络的参数进行较大的改动，而如果学习率过小，那么学习效率将会很低，算法很难收敛。因此学习率对于训练的效果有影响，是一个值得测试的部分。本文这里就是用了不同的学习率来实验，下面是实验设置和实验结果：

表 3 不同学习率的实验设置

编号	学习率	衰减因子	存储池	随机因子
1	0.01	0.9	10000	changed 0.6
2	0.04	0.9	10000	changed 0.6
3	0.09	0.9	10000	changed 0.6

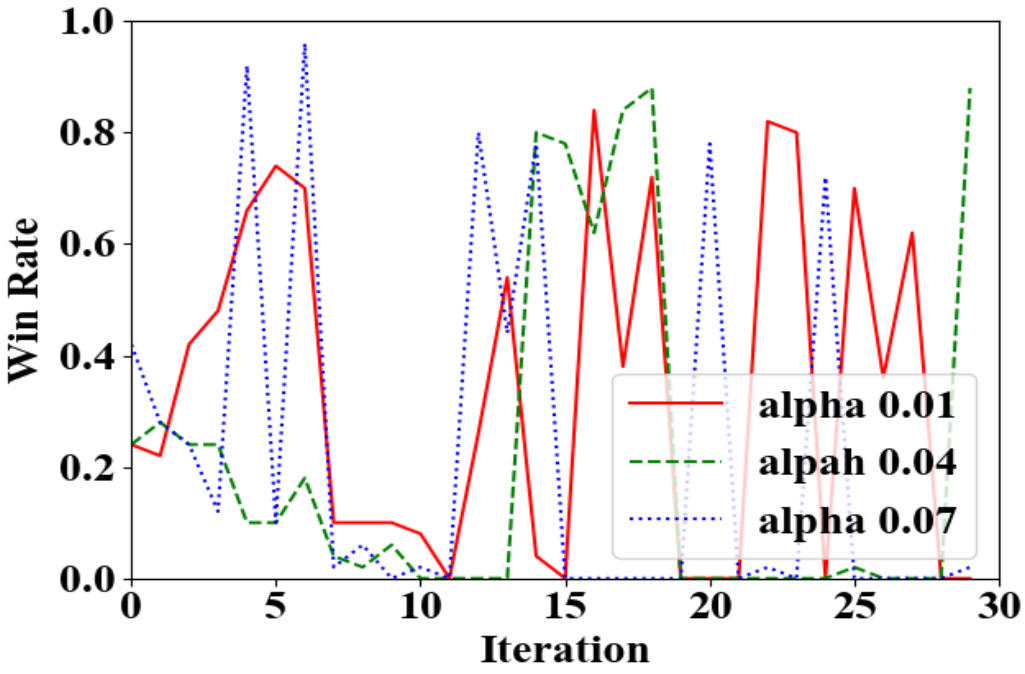

图 7 不同学习率效果对比

从实验数据来看，DQN 的效果并不理想，学习很不稳定，抖动性太大，无法最终达到一个较为稳定的胜率。从学习率的对波来看,可以发现不同的学习率对于 DQN 的表现还是有不小的影响的，学习率为 0.07 的实验中，峰值的胜率可达 96%，这个胜率还是非常好的，这说明学习率大学起来就快，但后续出现一些胜率的陡降，也说明了学习率大的话可能也会引进大的误差。

在 DQN 的算法中，衰减因子表示对于未来奖赏的衰减，体现的是更加看中当前步骤带来的奖赏还是对于未来步骤奖赏。如果衰减因子过大，那么对于当前步骤的奖赏就会更加看重，那么学习的效果就很大程度取决于当前步骤，由于样本是随机抽取的，那么当正样本比较密集的时候，学习效果会好。相反，如果衰减因子过小，那么就会提高对于未来奖赏的贡献比例，这样的学习会相对稳定，但是带来的问题就是提升慢，因为相当于对正的样本和负的样本都进行了平滑。下面是用不同的衰减因子进行实验设置和实验结果：

表 4 不同衰减因子的实验设置

编号	学习率	衰减因子	存储池	随机因子
1	0.01	0.85	10000	changed 0.6
2	0.01	0.9	10000	changed 0.6
3	0.01	0.95	10000	changed 0.6

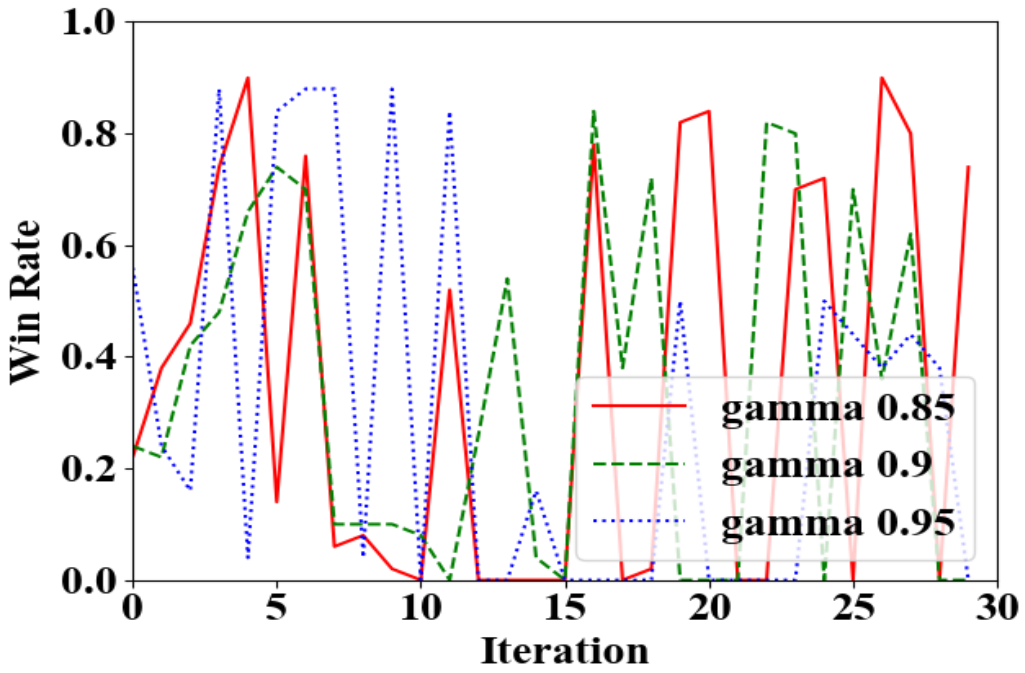

图 8 不同折扣因子效果对比

实验数据说明，采用不同的折扣因子也会带来不小的学习差异，如图中大的折扣因子在前期的训练拥有不错的效果，胜率能达到 90%左右，而中后期学习的效果反而是折扣因子小的更具有优势。可能的原因是前期样本池中随机的样本比较多，因此更加看重当前步的奖励，而后期样本池中训练得出来的样本较多，选择去依赖后续步骤的奖励效果相对较好。

DQN 之所以强大，其中一个原因是用到了经验回放，而经验回放需要一个存储池来存储样本，每次随机从中抽取样本，这样的好处是之前的样本有机会被多次学习，从而提升学习速度。而存储池的大小也是一个需要考虑的因素，如果太大，那么新样本被学习到的几率将会变小；如果太小，那么每个样本被重复学习的概率会变小，因此这也是一个值得去探究的因素。下面是用不同大小的存储池

进行实验设置和实验结果：

表 5 不同存储池的实验设置

编号	学习率	衰减因子	存储池	随机因子
1	0.01	0.9	4000	changed 0.6
2	0.01	0.9	10000	changed 0.6
3	0.01	0.9	20000	changed 0.6

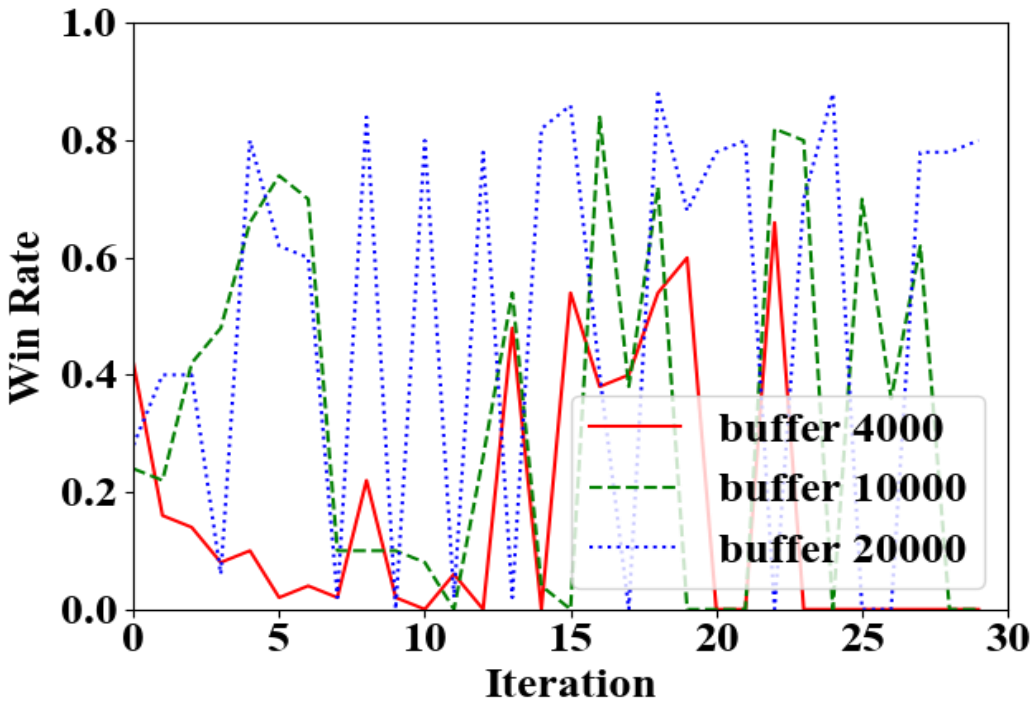

图 9 不同大小的存储池效果对比

这一组的实验数据说明，存储池的大小对于 DQN 算法的表现有重要影响。存储池越大，训练的效果就越好。这充分说明了 DQN 的经验回放的重要性，经验回放一方面用来切断经验数据的相关性，一方面可以做到样本的重复利用，最大化数据资源。因此，相对大的存储池能使得每次采样的数据会分布在更多的不同对局的游戏中，这样打破了数据关联性，学习效果会得到提升。但是越大的存储池的训练一般来说需要的资源开销会更大，因此也需要做一个权衡。

4.4.3 不同游戏难度下 DQN 的表现

这次本文训练的智能体对战的是 StarCraft2 中内置的电脑玩家，电脑玩家一共分 7 个难度，本文希望在不同的难度下进行实验，来评估 DQN 对于不同难度的电脑是否都有较好的适应能力。下面是对战不同难度的电脑的实验结果：

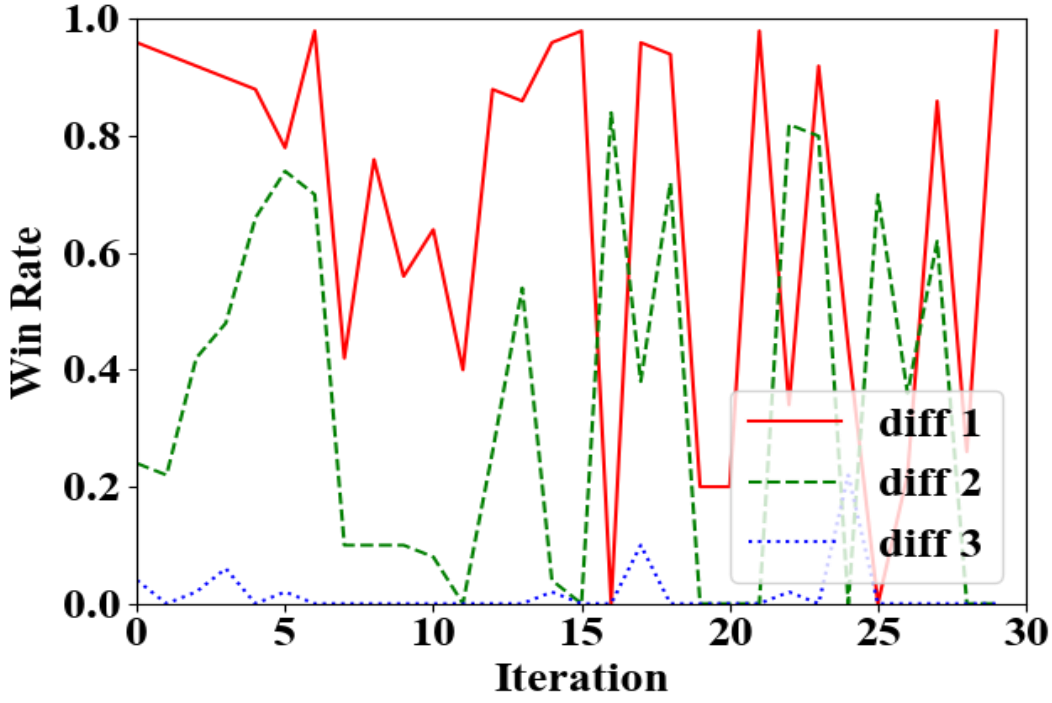

图 10 不同游戏难度效果对比

不同难度的游戏对于 DQN 的影响还是相当明显的，对战 1 级难度的内置电脑玩家，DQN 基本能达到 85%以上的胜率，而对战 3 级难度的内置电脑峰值胜率在 10%到 15%之间。这说明面对更加智能、进攻手段更加多样的高难度的电脑，DQN 就会不那么适用，无法去保持好的学习效果。

4.4.4 DQN 与 PPO 的表现对比

之前提到过，前人在 StarCraft2 中大多采用基于策略梯度的强化学习的方法，其中以 PPO 为典型代表。本文也希望以 DQN 和 PPO 为代表比较一下基于值函数的方法和基于策略梯度的方法在同一个游戏中的使用效果，以期对于后来在该

领域的研究者提供一些启发。下面是对比的实验结果：

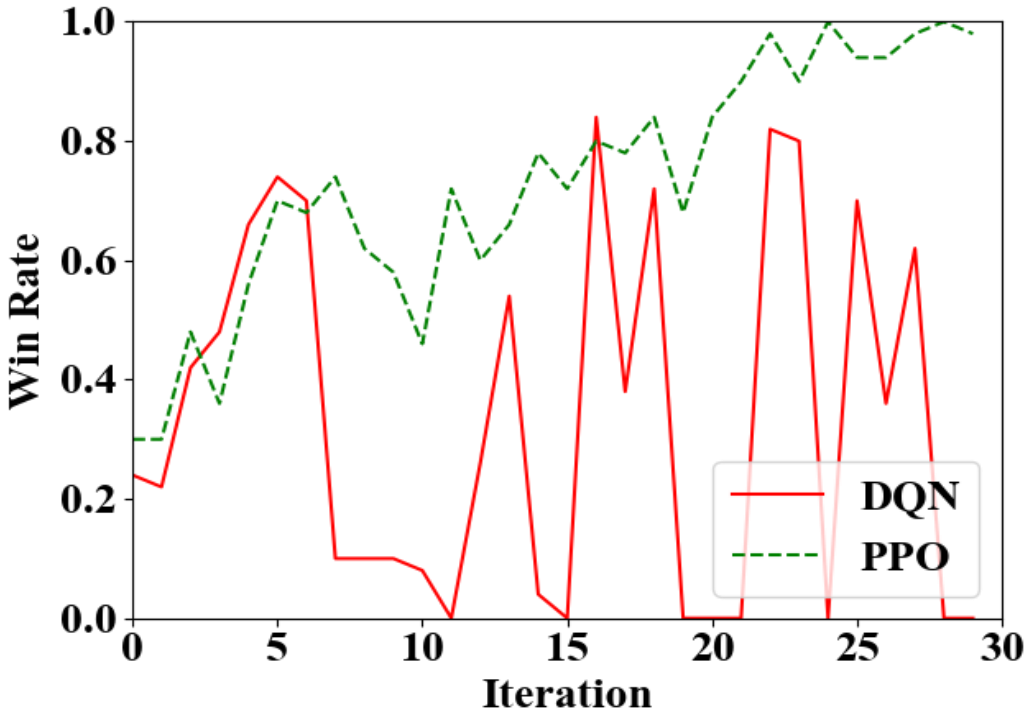

图 11 DQN 和 PPO 的效果对比

从实验结果来看，DQN 无法做到稳定的学习，胜率时而高、时而低，而 PPO 的效果虽然并不是完全单调地胜率上涨，但是总体上是不断上升的，甚至有时候能达到 100%的胜率，并且最终的胜率基本维持在一个较高的水平。这说明，对于 StarCraft2 这一款游戏来说，PPO 是更加适用的。DQN 不能说完全不适用，但是如果要有进一步的提升，还需要从多方面进行改进。

4.4.5 不同奖赏分配机制下 DQN 的表现

之前多次提到每一步的奖励是指导智能体去学习的重要因素，不同的奖励机制必然对于学习效果带来很大的影响。本文这里设计了两种不同点的机制来进行对照比较。一种是稀疏奖励，仅对结果设置奖励，具体到 StarCraft2 来说，前面所有步的奖励都是 0，仅有最后一步，根据对战的输赢给出奖励，输为-1，胜为 1，平为 0。第二种是富奖励，也就是游戏的每一步都会根据当前的状态给出

一个评分作为奖励，具体到 StarCraft2 来说，每一步根据当前的建筑物数量、士兵数量等等来给出奖励，从而指导智能体在生产建筑物和士兵的时候能够采取胜率更高的动作。下面是两种奖励机制的对比结果：

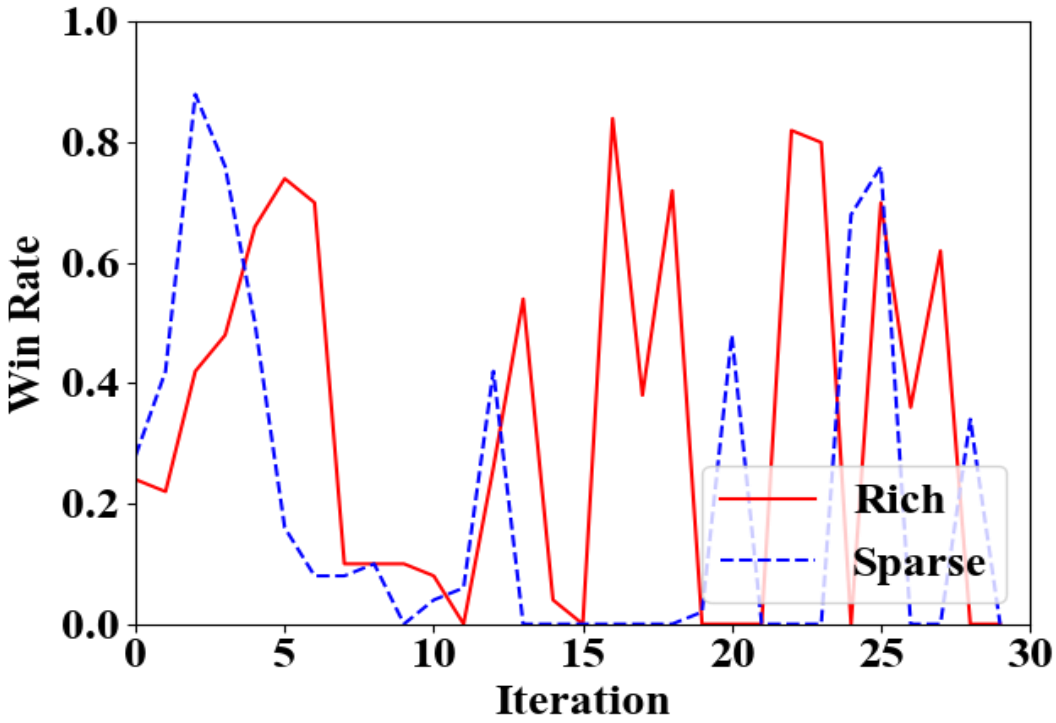

图 12 不同奖励机制效果对比

从实验结果看来，富奖励的效果相对来说更好，这说明如果每一步的奖励能具有差异性，对于 DQN 的指导效果会更好，但这样很多时候都会引入人类的一些先验知识，比如此处使用的建筑物数量、军队数量的信息很大程度带有人类的主观意愿，这些量的设定是从大量的对局中计算出来的。而如果使用输赢这样的稀疏奖励，好处是不需要人类的设定，但坏处是，除了最后一步其余奖励均是 0，对于 DQN 神经网络的参数更新没有明显的导向作用。因此，如何设计更好的奖励机制也是一个值得探究的方向。

4.4.6 Double DQN 在 StarCraft2 下的表现

在 DQN 的效果不够理想的情况下，本文又尝试了 DQN 的升级版 Double DQN，

Double DQN 的网络结构和 DQN 基本是一致的，唯一的不同是在计算当前状态的 Q 值的现实值的时候，采用了不同的计算方式。Double DQN 的好处是能解决 DQN 中存在的过估计问题。本文这里跑了 4 组实验，接下来是实验设置和实验结果：

表 6 Double DQN 的实验设置

编号	学习率	衰减因子	存储池	随机因子	奖励模式
1	0.01	0.9	20000	changed 0.9	rich
2	0.01	0.9	20000	changed 0.6	rich
3	0.01	0.9	5000	changed 0.6	rich
4	0.01	0.9	20000	changed 0.6	sparse

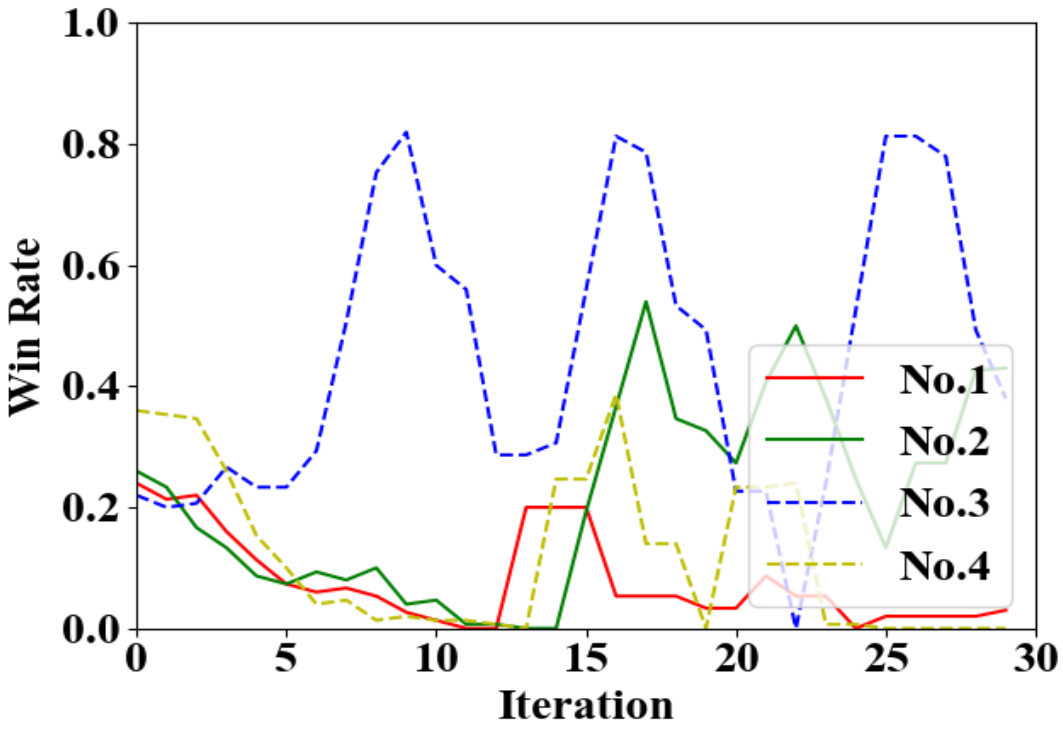

图 13 不同设置的 Double DQN 效果对比

通过不同的实验设置来看，各项参数对于 Double DQN 的应用效果也是有很大的影响的。从实验整体的结果来看，Double DQN 的效果是明显好于 DON 的，主要表现在由于解决了过估计问题，尽管还存在抖动，但是其抖动的幅度和频率都小于 DQN，其峰值胜率也不逊色于 DQN，因此 Double DQN 在一定程度上取得了

优于 DQN 的效果。

4.5 本章小结

本章分别在 CartPole 小游戏和 StarCraft2 游戏环境中进行了多组实验。在 CartPole 上使用 DQN 的效果非常好，可以看到游戏中，倒立摆能坚持不倒的时间是在快速稳定地增加的，这体现了 DQN 强大的学习能力。但是之后将 DQN 用在 StarCraft2 上时，效果就不太理想，具体表现在无法收敛到一个稳定的学习效果，游戏对战的胜率时而高时而低。本文从 DQN 算法本身的参数、游戏内置电脑的难度、奖励机制的设计等方面对改进实验效果做了努力，虽然峰值胜率可以到达一个理想的效果，但是胜率的抖动性还是表明 DQN 对于 StarCraft2 这类战略 RTS 游戏还是不太适用的。

第五章 总结和展望

5.1 本文总结

本文首先分析总结了深度强化学习邻域的基本的研究内容，接着介绍了相关邻域前人的工作，着重介绍了 Atari 游戏中使用的经典的 Deep Q-Network 算法，以及最新的在星际争霸 2 中用到的以 PPO 为代表的基于策略梯度的方法，并且总结了基于价值函数的深度强化学习方法和基于策略梯度的方法的各自的优点和缺点。鉴于目前在像星际争霸 2 这样的 RTS 游戏中大多都是采用基于策略梯度的方法，而很少使用值函数的方法，而 DQN 算法具有很强的通用性，本文想到去探究 DQN 在星际争霸 2 中应用的效果。在第三章本文叙述了所用到的实验方法，从搭建星际争霸 2 实验环境、对问题进行建模、修改 DQN 算法来适配星际争霸 2 环境等方面对实验的整个设计做了较为详细的说明。之后基于前文的理论，设计了一系列的对比对照实验来评估 DQN 在星际争霸 2 游戏中应用的效果。最终实验结果表明，DQN 在星际争霸 2 中的有一定的效果，但还有改进的空间。

除此之外，本文还尝试去探究不同给的奖赏分配机制对于 DQN 学习效果的影响，从而试图去加快学习的速度，提升学习的效率，降低学习的不稳定性。

5.2 未来研究展望

本文使用了当前深度强化学习研究者比较热衷使用的星际争霸 2 实验环境，成功在此环境中应用了 Deep Q-Network 算法来训练智能对战机器人，并且对对强化学习效果有重要影响的奖励机制进行探究，最终发现 DQN 对于星际争霸 2 这款游戏的使用效果还是没有那么理想。本文分析可能的原因有：算法本身参数设定过于依赖经验、纯粹使用建筑物、资源、军队等信息来表示状态可能还是不够、当游戏难度高了之后正样本难以获取、奖励的设计导向性不够或者不够客观、没有考虑对手信息等等。因此对于未来在星际争霸 2 上的研究者的启发有，可以考虑如何去自适应地调整参数的值、如何更好地表示游戏的状态信息、如何去设

计更适配的奖励机制、如何去获得高难度电脑下的正样本以及如何将对手的信息考虑进来，从而更好地去利用信息、从全局角度去评估不同决策的好坏。

参考文献

[1] Anschel O, Baram N, Shimkin N. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning[J]. 2016.

[2] Van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-learning[J]. Computer ence, 2015.

[3] Wang Z, Schaul T, Hessel M, et al. Dueling Network Architectures for Deep Reinforcement Learning[J]. 2015.

[4] Mnih Volodymyr, Kavukcuoglu Koray, Silver David, Rusu Andrei A, Veness Joel, Bellemare Marc G, Graves Alex, Riedmiller Martin, Fidjeland Andreas K, Ostrovski Georg, Petersen Stig, Beattie Charles, Sadik Amir, Antonoglou Ioannis, King Helen, Kumaran Dharshan, Wierstra Daan, Legg Shane, Hassabis Demis. Human-level control through deep reinforcement learning.[J]. Nature,2015,518(7540).

[5] Michael C. Fu. Simulation-Based Algorithms for Markov Decision Processes: Monte Carlo Tree Search from AlphaGo to AlphaZero. 2019, 36(06):25.

[6] Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. (2014). Deterministic policy gradient algorithms. In ICML, 2014.

[7] Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T, Lillicrap T, Silver D. Mastering Atari, go, chess and shogi by planning with a learned model. arXiv preprint arXiv:1911.08265, 2019.

[8] Jordan, M. I. and Mitchell, T. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245):255–260.

[9] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (2006). Learning representations by backpropagating errors. Nature 323, 533–536 (1986).

[10] Harm van Seijen, et al. (2017). Hybrid Reward Architecture for Reinforcement Learning. In NIPS, 2017.

[11] Shoham, Y., Powers, R., and Grenager, T. (2007). If multi-agent learning is the answer, what is the question? Artificial Intelligence, 171:365–377.

[12] Ni Zhen,Malla Naresh,Zhong Xiangnan. Prioritizing Useful Experience Replay for Heuristic Dynamic Programming-Based Learning Systems.[J]. IEEE transactions on cybernetics,2019,49(11).

[13] Risi, S., Preuss, M. From Chess and Atari to StarCraft and Beyond: How Game AI is Driving the World of AI. Künstl Intell 34, 7–17 (2020).

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. (2017) Proximal Policy Optimization Algorithms.

[15] Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. (2014). Deterministic policy gradient algorithms. In ICML, 2014.

[16] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In the International Conference on Learning Representations (ICLR).

[17] Martin Riedmiller. (2005). Neural Fitted Q Iteration—First Experiences with a Data Efficient Neural Reinforcement Learning Method. In ECML.

[18] .Russakovsky O, Deng J, Su H, et al. (2015). Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 2015, 115(3): 211-252.

[19] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. (2012). Proceedings of the 26th Annual Conference on Neural Information Processing Systems. Nevada, USA, 2012: 1097-1105.

[20] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In the International Conference on Learning Representations (ICLR).

[21] Mnih V , Kavukcuoglu K , Silver D , et al. Playing Atari with Deep Reinforcement Learning[J]. Computer Science, 2013.

致谢

这次实验工作的开展和论文的最终完成，离不开许多老师和同学方方面面的帮助。在这里向所有帮助、关心我的老师和同学们表示由衷的感谢！

首先感谢我的导师路通教授。从最初的选题，细心根据我的研究兴趣帮我筛选研究课题，拓展研究思路；中期实验，路通教授给了我许多实验方面的帮助和指导，使得我少走了许多弯路；最终论文撰写过程中，给了我很多规范化撰写论文的宝贵意见。路通教授的指导让我受益匪浅，不仅顺利完成了本次论文工作，也锻炼了我科研的能力，让我亲身体会到一个学术研究从开题到结题的整个过程，让我知道了如何去规范、严谨地做科研，受益终身。

我还要感谢计算机系俞扬老师对此次研究工作的支持。

此外，我要感谢计算机系刘若泽师兄对于我实验开展和论文写作中的帮助和建议。在论文开题后，刘若泽师兄帮助我搜集了相关的文献资料，给我的研究工作的规划提出了宝贵建议。同时，在论文写作中，尤其是后期论文修改过程中，刘若泽师兄都提出了很多专业、有建设性的修改意见。

最后，向耐心评阅我的论文的各位评审老师表示深深的感谢！