

题目内容:

自动驾驶技术研究在工业与民用领域都有着重要的应用。在自然环境与城市场景中，自动驾驶算法都需要能够实现对环境中障碍物的有效规避。深度强化学习方法能够通过与环境的交互训练网络模型与驾驶策略，从而得到具有规避障碍物功能的自动驾驶策略。研究需要根据应用场景的障碍物类型建立环境模型，设计深度强化学习模型，并对模型进行训练，得到所需的避障策略，最后设计自动驾驶避障算法，通过仿真验证算法的有效性。

任务要求:

1. 根据应用场景需求，建立环境模型。基于环境模型与任务要求，设计对应的价值函数、收益规则，设计深度强化学习模型；
2. 通过数据集对模型进行训练，并基于环境模型与深度强化学习模型训练得到能够实现避障功能的自动驾驶策略；
3. 自选编程语言与仿真平台，对训练得到的自动驾驶策略进行测试；
4. 将设计的自动驾驶避障策略与已有方法进行对比，进一步验证算法的有效性。

指导教师签字：_____

____年____月____日

教学单位负责人签字：_____

____年____月____日

责任教授签字：_____

____年____月____日

基于深度强化学习的自动驾驶避障技术研究

摘要

自动驾驶系统保证了快捷、安全、高效的驾驶体验，实现自动驾驶避障需要自动驾驶决策和控制系统的紧密配合。随着近些年硬件系统和软件算法的提升，自动驾驶系统在决策和控制方面的安全性和稳定性越来越受到广泛的关注。自动驾驶决策和控制系统的工作原理涵盖自动化技术、传感器技术、智能控制技术，也是人工智能和机器学习的重要应用领域。一直以来，实现自动驾驶的决策和控制系统需要构建大量的数学模型，依靠人工设计的算法从复杂环境中提取关键信息，因此降低自动驾驶决策和控制系统的复杂度一直是自动驾驶领域研究的重点方向之一。强化学习通过不断探索环境自主学习复杂的控制模型，深度学习与强化学习相结合形成的深度强化学习方法可实现端到端的决策与控制，逐渐成为自动驾驶领域的研究热点。

本文以实现端到端的自动驾驶决策器和控制器为研究目标，围绕两种不同复杂度的仿真环境，针对 DQN 及其改进算法搭建神经网络模型，验证实现了基于 DQN 及其改进算法的自动驾驶决策器和控制器，并就 DQN 及其改进算法开展了较为深入的研究。

实验结果表明，本文设计实现的基于 DQN 及其改进算法的自动驾驶决策器和控制器均满足实验要求，达到了预期的控制效果，能够有效提高自动驾驶车辆在决策和控制中的鲁棒性。对于较为简单的网络结构，DQN 特别是 Double DQN 算法由于其改进的动作选择和评估方法，能够获得更加稳定有效的行为策略。本文的成果为自动驾驶决策器和控制器的研究提供了借鉴和参考，也为复杂动力学模型问题的解决提供了新的思路和方法。

关键词：自动驾驶避障；自动驾驶决策；控制器设计；DQN 网络；深度强化学习；端到端驾驶

Research on Automatic Driving Obstacle Avoidance Based on Deep Reinforcement Learning

Abstract

The Automatic Driving System ensures a fast, safe and efficient driving experience. The implementation of Automatic Driving Obstacle Avoidance requires the cooperation of the decision-making and control system of Automatic Driving. As the hardware and software improves, the safety and stability of such systems have received more attention. The design of these systems involves Automation Technology, Sensor Technology, Intelligent Control Technology, which is also an important field of Artificial Intelligence and Machine Learning. It has always been necessary to build a large number of mathematical models to realize such systems, which also rely on artificially designed algorithms to extract key information from complex environments. Therefore, reducing the complexity of these systems has always been the focus of research in the field of Autonomous Driving. Reinforcement Learning learns complex control models autonomously by exploring the environment continuously. The Deep Reinforcement Learning method, formed by the combination of Deep Learning and Reinforcement Learning, can realize end-to-end decision-making and control, and has gradually become a research hotspot in the field of Autonomous Driving.

This work aims to implement the end-to-end Autonomous Driving decision maker and controller. Focusing on two simulation environments with different complexity, a neural network model is built for DQN and its improved algorithms. The performance is verified. This paper also conducts more in-depth research on DQN and its improved algorithms.

The experimental results show that the Autonomous Driving decision maker and controller based on DQN and its improved algorithms designed and implemented in this paper meet the experimental requirements, achieve the expected control effect, and can effectively improve the robustness of autonomous driving vehicles in decision-making and control. For simpler network structures, DQN, especially the Double DQN algorithm, can

obtain more stable and effective behavior strategies due to its improved action selection and evaluation methods. The results of this paper provide reference for the research of Automatic Driving decision maker and controller, and also provide new ideas and methods for solving complex dynamic model problems.

Key Words: Automatic Driving Obstacle Avoidance; Automatic Driving Decision-Making; Controller Design; DQN Network; Reinforcement Learning; End-to-end Driving

目 录

摘要	I
Abstract	II
第1章 绪论	1
1.1 研究背景与意义	1
1.2 国内外研究现状	2
1.3 本文主要研究内容及章节安排	5
第2章 强化学习相关算法理论分析	6
2.1 强化学习算法	6
2.2 Q-learning 算法	9
2.3 DQN 算法	10
2.4 改进的 DQN 算法	13
2.4.1 Double DQN 算法	14
2.4.2 Dueling DQN 算法	14
2.5 本章小结	16
第3章 基于 DQN 算法的自动驾驶避障算法设计	17
3.1 基于 DQN 算法的决策器设计	17
3.1.1 仿真环境	17
3.1.2 网络结构设计	19
3.1.3 决策器的学习过程	20
3.2 基于 DQN 算法的控制器设计	21
3.2.1 仿真环境	21
3.2.2 网络结构设计	24
3.2.3 控制器的学习过程	25
3.3 本章小结	26
第4章 仿真实验验证	27
4.1 基于 DQN 算法的决策器实验	27
4.2 基于 DQN 算法的控制器实验	31
4.3 本章小结	34
结 论	35
参考文献	36

第 1 章 绪论

1.1 研究背景与意义

自动驾驶系统的决策模块需要先进的决策算法保证安全性、智能性、有效性。目前传统算法的解决思路是以价格昂贵的激光雷达作为主要传感器，依靠人工设计的算法从复杂环境中提取关键信息，根据这些信息进行决策和判断。该算法缺乏一定的泛化能力，不具备应有的智能性和通用性。深度强化学习的出现有效地改善了传统算法泛化性不足的问题，这给智能驾驶领域带来新的思路。

强化学习 (Reinforcement Learning, RL) 通过与环境交互，学习状态到行为的映射关系。如图1-1所示，在一个离散时间序列 $t = 0, 1, 2, \dots$ 中，智能体需要完成某项任务。在每一个时间 t ，智能体都能从环境中接受一个状态 S_t ，并通过动作 a_t 与环境继续交互，环境会产生新的状态 S_{t+1} ，同时给出一个立即回报 r_t 。如此循环下去，智能体与环境不断地交互，从而产生更多数据（状态和回报），并利用新的数据进一步改善自身的行为。

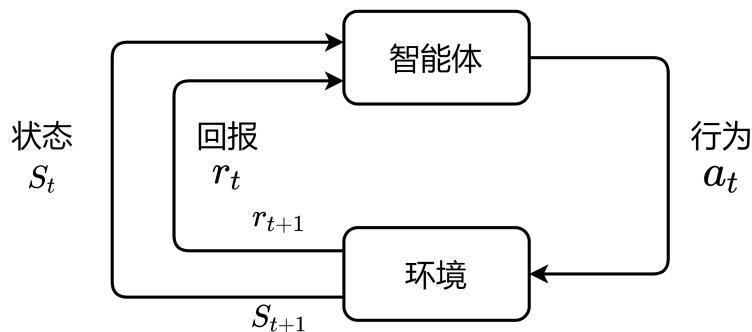


图 1-1 强化学习原理

目前，强化学习在策略选择的理论和算法方面已经取得了很大的进步，然而直接从高维感知输入（如图像、语音等）中提取特征，学习最优策略，对强化学习来说依然是一个挑战。

深度强化学习 (Deep Reinforcement Learning, DRL) 结合了深度神经网络和强化学习的优势，可以用于解决智能体在复杂高维状态空间中的感知决策问题^{[1][2]}。2016 年，基于深度强化学习和蒙特卡洛树搜索的 AlphaGo 击败了人类顶尖职业棋手，引起了全世界的关注^[3]。2017 年，DeepMind 在《Nature》上公布了最新版 AlphaGo 论

文，介绍了更强的围棋人工智能：AlphaGo Zero。它不需要人类专家知识，只使用纯粹的深度强化学习技术和蒙特卡罗树搜索，经过 3 天自我对弈就以 100 比 0 击败了上一版本的 AlphaGo。AlphaGo Zero 证明了深度强化学习的强大能力，也必将推动以深度强化学习为代表的人工智能领域的进一步发展。基于深度强化学习在棋局与游戏上的成功，最近的研究大多注重于深度强化学习在各个领域中的扩展与应用。

综上所述，深度强化学习方法与深度神经网络强大的特征提取能力相结合，可以实现端到端的控制与决策，具有较强的通用性。通过网络自主学习的方式，减少了对系统动力学建模与数学解析的复杂度，相比于传统依据规则的决策方式更加便捷。随着人工智能的兴起和强化学习在轮式机器人相关领域的成功应用，基于深度强化学习的自动驾驶决策与控制方法为自动驾驶决策提供了新的解决方案，这使得对其研究更具有理论指导意义和实际应用价值。

1.2 国内外研究现状

自动驾驶系统 (Automated Driving Systems, ADS) 保证了安全、舒适和高效的驾驶体验，但近年来的研究表明，除非进一步提高最新技术的鲁棒性，否则自动驾驶系统的潜力便无法完全发挥^[4]。目前，大多数的自动驾驶系统将大量的自动驾驶任务划分为若干个子类别，并在各个模块上采用一系列传感器和算法，算法流程如图1-2所示。

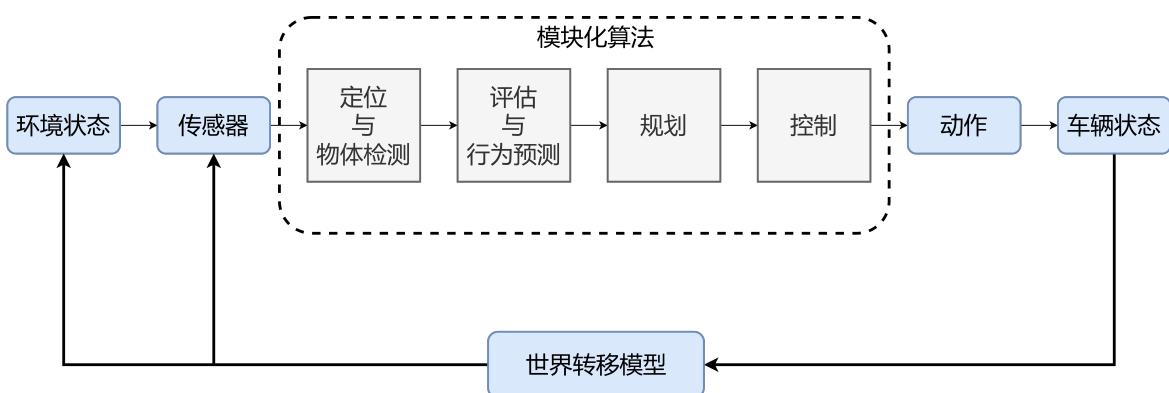


图 1-2 模块化方法流程图

最近，端到端方法开始作为模块化方法的替代出现。端到端驾驶 (End-to-end Driving) 又被称作直接感知 (Direct Perception)^[5]，即直接从感知输入产生动作，其算法流程图如图1-3所示。此处的动作可以是方向盘和踏板的连续操作，也可以是一组

离散的动作，例如加速和转向。目前有三种主要的端到端方法：直接监督深度学习^{[6][7]}、神经进化^[8](Neuroevolution) 和深度强化学习^[9]。

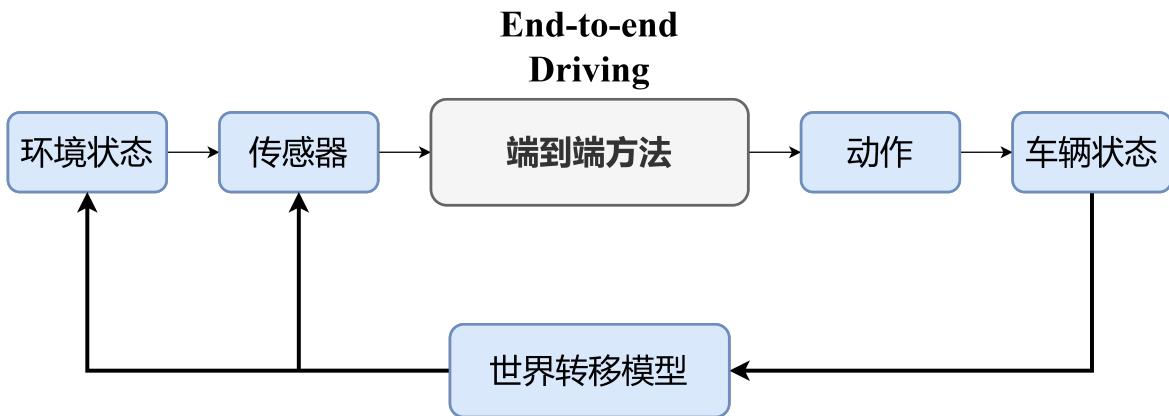


图 1-3 端到端方法流程图

深度学习和强化学习的发展使得直接从原始的数据中提取高水平特征进行感知决策变成可能。深度学习起源于人工神经网络。早期研究人员提出了多层感知机的概念，并且使用反向传播算法优化多层神经网络，但是由于受到硬件等资源的限制，神经网络的研究一直没有取得突破性进展。最近几年，随着计算资源的性能提升和相应算法的发展，深度学习在人工智能领域取得了一系列重大突破，包括图像识别、语音识别、自然语言处理等。深度学习由于其强大的表征能力和泛化性能受到众多研究人员的关注，相关技术在学术界和工业界都得到了广泛的研究与应用。

深度强化学习由数据驱动，不需要构造系统模型，具有很强的自适应能力。普林斯顿大学的 Chen 等使用深度学习算法，根据摄像头采集的图像数据预测目标的距离，同时输出操作指令^[5]。斯坦福大学的 Zhu 等使用暹罗网络结构，同时输入当前视角图像和目标物体图像，并且使用残差网络模型提取特征。通过 A3C (Asynchronous Advantage Actor-critic) 算法进行训练，成功控制小车在虚拟场景和现实场景中到达指定地点^[10]。国内的 Zhao 等使用深度强化学习算法和注意力机制，实现了智能驾驶领域车辆的高精度分类^[11]。Zhu 基于 TORCS 的真实物理变量，使用高斯过程强化学习算法 PILCO (Probabilistic Inference for Learning Control) 离线训练控制器，实现车道保持。同时以图像为输入，使用深度学习算法感知环境信息，预测本车距离车到中央线距离、偏航角、道路曲率等。最终将 RL 的控制策略和 DL 的特征预测结合，实现基于图像的车道保持。

以上研究仅由大量的数据驱动而未单独考虑模型的影响，针对自动驾驶避障技术的模型问题，德克萨斯大学奥斯汀分校的 Chen 等提出“Learning to drive from a world on rails”的假设^[12]，即车辆运动只与自身模型相关而与世界模型无关，采用世界模型与车辆模型解耦的方式建立一个前向模型来帮助改进自动驾驶策略；David Ha 等提出生成循环神经网络以无监督的方式快速训练，通过压缩时空表征来模拟流行的强化学习环境^[13]。

针对具体的深度强化学习算法，2012 年，Lange 等人最早在插槽赛车上使用深度强化学习算法并取得了良好的控制效果^[14]。2013 年，由 DeepMind 团队提出的 DQN (Deep Q-Network) 算法利用深度卷积神经网络直接学习 Atari2600 种游戏的高维度图像，从输入中提取环境的高效描述，来近似最优动作-状态函数，从而习得成功策略^[15]。2015 年，Hasselt 等人发现传统的 Q-learning 和 DQN 方法都会普遍过高估计行为值函数 Q 值，存在过优化的问题，为了解决值函数过估计的问题，Hasselt 提出了 Double DQN 方法，将行为选择和行为评估采用不同的值函数实现，降低了过估计的误差^[16]。2016 年，Wang 提出 Dueling DQN 方法，把 Q 网络的结构显式地约束成跟动作无关的状态值函数 $V(s)$ 与在状态 s 下各个动作的优势函数 $A(s, a)$ 之和，使得 DQN 训练更容易，收敛速度更快，避免因为 Q 值的量级大而引起的结果不稳定问题^[17]。2017 年，Zong 等人使用深度确定性策略梯度 DDPG 算法对智能体的加速度和转向控制进行训练，以实现自主避障，并在开源赛车模拟器 (TORCS) 环境中进行了测试，结果表明通过一段时间自主学习，无人驾驶汽车能学会复杂的避障换道行为^[18]。该算法是将 DQN 和 Actor-Critic 算法相结合，使用深度神经网络来逼近值函数和策略函数，虽然，DDPG 算法实现了连续状态、动作空间下的强化学习，然而算法参数并不易确定，其超参数往往必须针对不同的问题进行仔细设置才能获得良好的训练结果。

目前，一般的深度强化学习在商用领域仍未落地，当前的算法大多是采用深度强化学习进行端到端的路径规划，通过传感器输入的信息和先验地图信息输出轨迹信息，百度 Apollo 团队在 2022 年发布的 ApolloRL 平台^[19]，采用 DRL 作为自动驾驶车辆的轨迹输出，并用多种控制器如 (MPC、LQR、PID) 等对路径实现紧密跟踪，百度团队的做法对于深度强化学习在自动驾驶中的落地具有一定的借鉴作用。

随着研究的不断深入，深度强化学习算法在自动驾驶控制策略的应用范围越来越广，解决的问题也越来越复杂。这充分说明了深度强化学习算法应用于自动驾驶

决策控制领域的可行性和有效性。本文在调研和分析相关文献的基础上基于深度强化学习算法设计自动驾驶决策控制算法，通过对比多种深度强化学习算法及其改进算法，验证在抽象环境与高维环境下深度强化学习算法的有效性和准确度。

1.3 本文主要研究内容及章节安排

本文以 DQN 算法作为基础，探究 DQN 及其改进算法 Double DQN 和 Dueling DQN 在两种不同仿真环境（Highway-Env、Metadrive）中对自动驾驶决策及控制产生的效果。通过在仿真环境下的反复训练和学习，训练的模型在多种仿真环境下均具有较好的鲁棒性，证明了 DQN 及其改进算法在自动驾驶决策及控制方面的有效性和可行性。

第 1 章绪论。介绍了该课题的研究背景和意义，对强化学习在自动驾驶决策方面的发展现状进行了介绍，对深度强化学习特别是 DQN 算法的研究现状进行了详细的分析说明，在章节的最后通过分析总结参考文献得出了本文的研究方向和研究内容。

第 2 章强化学习相关算法理论分析。通过对强化学习算法原理的分析，引出了适用于无模型的 Q-learning 算法和 DQN 算法，同时，针对于 DQN 算法的不足和缺陷，介绍了 Double DQN 算法和 Dueling DQN 算法的原理，并对几种算法的适用场景进行了分析。

第 3 章基于 DQN 算法的自动驾驶避障算法设计。通过对两种仿真环境（Highway-Env、Metadrive）的分析和对比，介绍二者的状态值、动作值与奖励算法的详细情况。将 DQN 及其改进算法作用于两种仿真环境，同时改进 Q 网络的网络结构及超参数，使其完成自动驾驶决策和控制两方面的任务要求。

第 4 章实验验证。为了验证算法的可行性和有效性，进行了两种仿真环境中的仿真实验，并对具体的实验内容进行了分析与设计，对最后的实验结果进行了详细的分析与改进。

文章的最后进行了总结，对本文完成的主要工作和取得的主要成就进行了概述，对本文的创新点进行了阐述，同时指出了本文研究的不足和后续研究的重点内容。

第 2 章 强化学习相关算法理论分析

如图1-1所示，强化学习的基本原理在1.1节中已有提及，在此不再赘述。强化学习强调智能体与环境不断地交互，从而产生更多数据（状态和回报），并利用新的数据进一步改善自身的行为。智能体不会被告知在当前状态下，应该采取哪一个动作，只能通过不断尝试，依靠环境对动作的反馈改善自己的行为。经过数次迭代后，智能体最终能学到完成相应任务的最优动作（策略）。

本章通过对强化学习以及相关算法的理论分析，对基于值函数（Value Based）的 Q-learning 算法及 DQN 算法进行详细介绍，对 DQN 算法及其改进算法的优劣进行比较，选择最适合解决自动驾驶决策控制的方法。

2.1 强化学习算法

强化学习包括智能体和环境两大对象。智能体又称为学习者或玩家，环境是指与智能体交互的内部。智能体由策略、值函数、模型三个组成部分中的一个或多个组成。下文将介绍强化学习智能体的各个组成部分与强化学习问题求解的目标，由此引出基于值函数的强化学习方法。

强化学习算法的组成

(1) 策略：策略是决定智能体行为的机制，是状态到行为的映射，用 $\pi(a|s)$ 表示，它定义了智能体在各个状态下的各种可能的行为及概率。

$$\pi(a|s) = P(A_t = a|S_t = s) \quad (2-1)$$

策略分为两种，确定性策略和随机性策略。确定性策略根据智能体具体状态输出一个确切的动作，而随机性策略根据状态输出智能体每个动作的概率，输出值为一个概率分布。一个策略完整定义了智能体在各个状态下的各种可能的动作及其概率大小。策略仅和当前状态有关，与历史信息无关。策略就是用来描述各个不同状态下执行各个不同行为的概率，同一时刻某一确定的策略是静态的，与时间无关，但是智能体可以随着时间更新策略。

(2) 值函数：值函数代表智能体在给定状态下采取某个行为的好坏程度。这里的

好坏用未来的期望回报表示，而回报和采取的策略有关，所有值函数的估计都是基于给定的策略进行的。值函数（或称为回报）用 G_t 表示，也称为“收益”或“奖励”。

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (2-2)$$

其中折扣因子 γ （衰减系数）体现了未来的回报在当前时刻的价值比例，在 $k+1$ 时刻获得的回报 R 在 t 时刻体现出的价值是 $\gamma^k R$ 。 γ 接近 0 表示倾向于当前利益； γ 接近 1 表示偏向于长远期的利益。

值函数分为状态值函数与状态行为值函数，二者都与回报有关。

状态值函数 $V_\pi(s)$ 表示从状态 s 开始，遵循当前策略 π 所获得的期望回报。

$$\begin{aligned} V_\pi(s) &= E_\pi[G_t | S_t = s] \\ &= E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \end{aligned} \quad (2-3)$$

值函数的另一个类别是状态行为值函数 $Q_\pi(s, a)$ ，也称为行为值函数。该函数表示针对当前状态 s 执行某一具体行为 a 后，继续执行策略 π 所获得的期望回报；也表示遵循策略 π 时，对当前状态 s 执行行为 a 的价值大小。

$$\begin{aligned} Q_\pi(s, a) &= E_\pi[G_t | S_t = s, A_t = a] \\ &= E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a] \end{aligned} \quad (2-4)$$

(3) 模型：在强化学习任务中，模型是智能体对环境的一个建模。环境模型至少要解决两个问题，一是预测状态转移概率 $P_{ss'}^a$ ，即预测在状态 s 上采取行为 a 后，下一个状态 s' 的概率分布；二是预测在状态 s 上采取行为 a 后可能获得的立即回报 R_s^a 。

$$\begin{aligned} P_{ss'}^a &= P(S_{t+1} = s' | S_t = s, A_t = a) \\ R_s^a &= E[R_{t+1} | S_t = s, A_t = a] \end{aligned} \quad (2-5)$$

根据智能体在与环境交互的过程中是否建立环境的模型，强化学习可以分为两个大类，即有模型方法和无模型方法。一般的模型已知问题，就是智能体获得了确切的状态转移概率 $P_{ss'}^a$ 和回报 R_s^a 。

在后续章节的仿真环境介绍中，由于车辆的决策控制较难采用已知模型（如动

力学模型、运动学模型)进行刻画,故均使用“无模型方法”的假设,即智能体在整个训练过程中不需要对环境模型进行建模,直接使用学习得到的经验进行策略的优化。

根据以上三点概念,可以通过建立状态值估计的方法或建立策略估计的方法来解决强化学习问题。基于值函数的方法在求解强化学习的目标时只估计状态值函数,不估计策略函数,最优策略函数在对值函数进行迭代求解时,通过状态值函数间接得到。针对自动驾驶避障问题,由于我们较难估计车辆状态与行为之间的映射(即策略函数),但可以估计车辆在采取某个动作时,车辆状态发生的变化和环境回报(即状态值函数),所以采取基于值函数的方法更加符合解决自动驾驶避障问题所需要达成的目标。

强化学习算法的求解

求解强化学习问题的目标是求解每个状态下的最优策略,即在运行过程中接收的累计回报最大。为了获取更高的回报,智能体在进行决策时要考虑立即回报,也要考虑后续状态的回报。解决强化学习问题一般需要两步,将实际场景抽象成一个数学模型,然后去求解这个数学模型,找到使得累计回报最大的解。

第一步:构建强化学习的数学模型——马尔科夫决策(Markov Decision Process, MDP)^[20]模型。

不论涉及的智能体结构、环境和交互细节多么复杂,此类交互问题都能简化为三个信号:智能体的行为、环境的状态、环境反馈的回报。具体到实验中,便是仿真环境根据智能体做出的行为产生的回报和改变的状态。马尔科夫决策模型可以有效表示实际的强化学习问题,这样解决强化学习问题的问题就转化为求解马尔科夫决策模型的最优解。

第二步:求解马尔科夫决策模型的最优解。

求解马尔科夫决策问题,是指求解每个状态下的行为,使得累计回报最大。对于环境已知的情况可以选用基于模型的方法如动态规划法;基于未知的情况选择无模型方法如时序差分法;对于状态空间、动作空间连续的场景可以采用值函数逼近法等等。根据上一小节的分析,求解马尔科夫决策模型的方法便是构建具体的算法与网络结构,下文将对基于值函数(Value Based)的Q-learning算法及DQN算法进行详细介绍。

2.2 Q-learning 算法

在介绍 Q-learning 算法之前，首先对时序差分方法进行简单的介绍。时序差分学习最早由 A.Summuel 在跳棋算法中提出，1988 年，Sutton 证明了时序差分方法在最小均方误差（Mean Square Error, MSE）上的收敛性^[21]，之后时序差分方法被广泛应用在无法产生完整轨迹的无模型强化学习问题上。时序差分（TD）方法是无模型方法，无法获得当前状态的所有后续状态及回报，仅能通过采样学习轨迹片段，用下一状态的预估状态价值更新当前的状态价值。

Q-learning 算法属于离线策略时序差分（TD）问题，最早由 Watkins 和 Dayan 在 1992 年提出^[22]，其任务是通过不断地学习，不断的更新状态-动作值函数 $Q(s, a)$ ，从而得出最优策略。根据 2.1 节中所提到的，求解强化学习问题的目标是求解每个状态下的最优策略，Q-learning 算法在更新一个状态-动作值函数（以下简称 Q 值）时，采用的不是遵循当前策略（行为策略 μ ）的下一个状态-动作对的 Q 值，而是待评估策略（目标策略 π ）产生的下一个状态-动作对的 Q 值。更新公式如下：

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t)) \quad (2-6)$$

其中 α 称为学习率， γ 称为折扣因子，TD 目标 $R_{t+1} + \gamma Q(S_{t+1}, A')$ 是基于目标策略 π 产生的行为 A' 得到的 Q 值和一个立即回报的和。在 Q-learning 算法中，行为策略 μ 是基于原始策略的 ϵ -贪心策略，保证取得经历足够丰富的新状态。目标函数 π 是单纯的贪心策略，通过最大化 TD 目标来保证策略最终收敛到最佳策略。

Q-learning 算法处理有限的状态空间与有限的动作空间的问题，状态值和动作放在 Q 表中，值函数能够表示为一个数组。但在实际情况下，强化学习面临的问题的状态空间往往是连续的，无法用表格的方法准确列出每一种状态对应的 Q 值大小，故需要进行对 Q 值进行非线性的逼近。下文的 DQN 就属于这样的方法。无论是 Q-learning 或是 DQN，均采用了目标策略 π 产生的下一个状态-动作对的 Q 值对原有的 Q 值进行更新，这是时序差分法的一般思想。

Q-learning 的算法流程如下：

Algorithm 1: Q-learning 算法

Input: 环境 E , 状态 S , 动作 A , 折扣因子 γ , 学习率 α , 初始化行为值函数 $Q(s, a) = 0$

for $k = 0, 1, 2, \dots, m$ **do**

初始化状态 s ;

for $t=0,1,2, \dots$ **do**

在 E 中通过 π 的 ϵ -贪心策略采取行为 a ;

$r, s' =$ 在 E 中执行动作 a 产生的回报和转移的状态;

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$;

$s \leftarrow s'$;

end

end

$\pi^*(s) = \arg \max_{a \in A} Q(s, a)$;

Output: 最优策略 π^*

2.3 DQN 算法

DQN (Deep Q-Network) 算法是建立传统强化学习算法 Q-learning 的基础上的时序差分算法, Q-learning 是离线策略时序差分法, 使用 ϵ 贪心策略产生数据, 利用查表法对行为值函数 (Q 值) 进行预测, TD 目标是 $R_{t+1} + \gamma Q(S_{t+1}, A')$ 。DQN 算法在传统强化学习 Q-learning 的基础上, 主要对其精确的查表法做了近似拟合, 同时通过引入深度学习网络, 对网络结构和参数更新做出了如下改进。

(1) DQN 使用深度神经网络从原始数据中提取特征, 近似行为值函数 (Q 值)。

当状态空间很大且连续时, 无法使用查表法来求解每个状态的价值, 此时可以考虑“离散”状态空间的方法来减少算力。在“离散”状态空间中, 使用深度神经网络来表示行为值函数是常见的方法。对于深度神经网络, 其参数是每层网络的权重及偏置, 用 θ 表示, 对值函数的更新等价于对参数 θ 的更新。DQN 神经网络结构如图2-1所示。

DQN 神经网络结构是三个卷积层和两个全连接层。输入为经过处理的 4 个连续的 84×84 的图像, 经过卷积层和两个全连接层输出包含每一个动作的 Q 值向量。DQN 网络将高维的状态输入转换为低维的动作输出, 即将图像输入转换为动作输出。

利用深度神经网络实现了数据的降维。

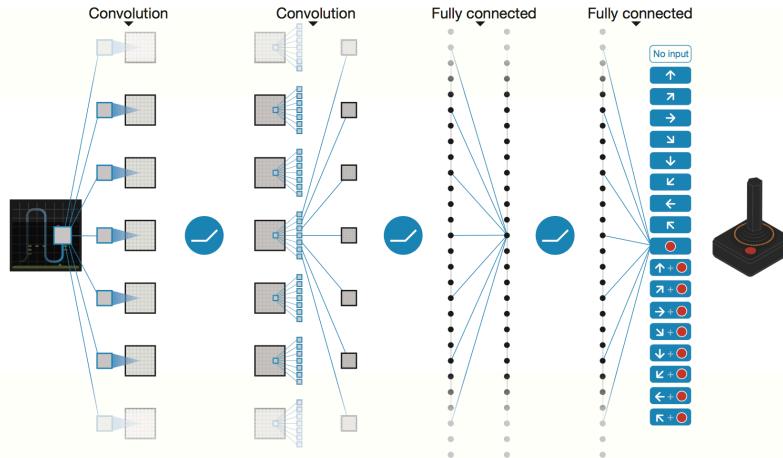


图 2-1 DQN 神经网络结构^[15]

(2) DQN 使用经历回放训练强化学习。

在使用深度神经网络进行行为值函数（Q 值）近似时，如果不对训练数据做处理，直接将当前时刻的信息进行学习训练，学习效果会出现较大偏差。由于使用神经网络的前提是数据之间独立同分布，而强化学习过程中，数据是通过与环境交互产生的，相邻数据之间高度相关。如果智能体在很长一段时间均学习相同环境下的数据，在接收到另一环境的数据后，参数会出现不稳定与大范围波动发散，求解无法收敛。针对这一问题，DQN 采用“经验回放”的方法进行解决。

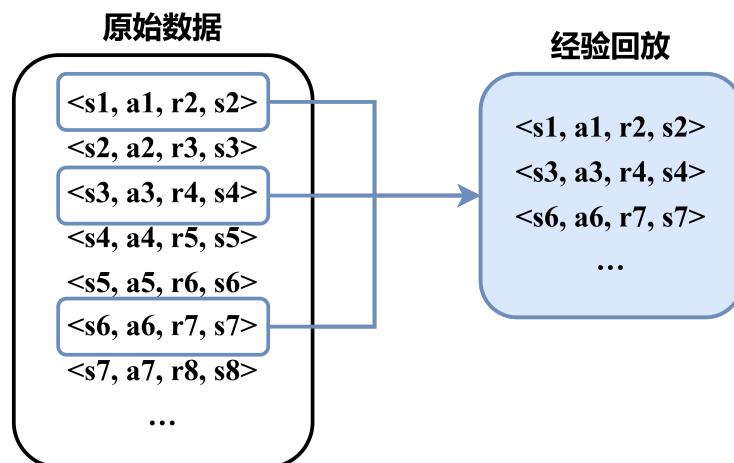


图 2-2 经验回放

经验回放最早由 Long Ji Lin 在 1993 年提出^[23]，如图2-2，它在强化学习中是这样实现的：智能体跟环境不断交互，将在环境中积累的数据存储到记忆库中。首先对环境做出探索并将 <本时刻状态、行为、奖励、下一时刻状态> (< s_1, a_1, r_2, s_2 >) 作为一个事件对进行存储，每一次对神经网络中的参数进行更新时，利用均匀随机采样的方法从数据库中抽取数据，通过抽取的数据对神经网络进行训练。因为经验回放的样本是随机抽取，每次用于训练的样本不再是连续的数据，打破了数据的关联，由此可以满足神经网络的假设要求。

(3) DQN 使用单独的目标网络处理 TD 偏差。

与式2-6类似，DQN 更新神经网络的参数 θ 采用的是梯度下降法。更新公式如下：

$$\theta_{t+1} = \theta_t + \alpha(r + \gamma \max_{a'} Q(s', a'; \theta_t) - Q(s, a; \theta_t)) \nabla Q(s, a; \theta_t)$$

但如 (2) 中提及，行为值函数 ($Q(s, a; \theta_t)$) 和 TD 目标值函数 ($r + \gamma \max_{a'} Q(s', a'; \theta_t)$) 产生的数据需要避免关联性。为解决上述问题，DQN 引入两个神经网络，一个网络固定参数专门用来产生 TD 目标，称为 TD 网络。另一个网络专门用来评估策略更新函数，逼近值函数，称为行为值函数 (Q 值) 逼近网络。两个网络参数的更新速率不一致，用于行为值函数 (Q 值) 逼近的网络参数每一步都更新；用于计算 TD 目标值的网络参数每隔固定的步数更新一次，期间保证不变。于是得到以下 DQN 网络的更新公式：

$$\theta_{t+1} = \theta_t + \alpha(r + \gamma \max_{a'} Q(s', a'; \theta_t^-) - Q(s, a; \theta_t)) \nabla Q(s, a; \theta_t) \quad (2-7)$$

综合上述三点改进，DQN 算法将经验回放和设置单独的目标网络两个方面对 Q-learning 方法进行改善，使其对于更加复杂的问题和大规模神经网络更加稳定和容易收敛，DQN 的算法流程如下：

Algorithm 2: DQN 算法

Input: 环境 E , 状态 S , 动作 A , 折扣因子 γ , 学习率 α

初始化经验回放库 D 并定义容量 N ;

随机初始化网络参数 θ , 用 θ 初始化主网络 $Q(\cdot; \theta)$;

随机初始化网络参数 $\theta^- = \theta$, 用 θ^- 初始化 TD 网络 $Q(\cdot; \theta^-)$;

for $k = 0, 1, 2, \dots, m$ **do**

 初始化状态 s ;

for $t=0, 1, 2, \dots$ **do**

 在 E 中通过主网络的 ϵ -贪心策略采取行为 a (以 ϵ 概率随机选择任一随机动作, 以 $1 - \epsilon$ 概率选择行为值函数最大的动作, 即

$$a = \arg \max_{a \in A} Q(s, a; \theta);$$

 在 E 中执行动作 a , 返回奖励 r , 和下一时刻状态 s' ;

 将当前事件对 $< s, a, r, s' >$ 存入经验回放库 D 中;

 从经验回放库 D 中随机采样 n 个数据, 进行如下算法更新:

$$q_{target} = \begin{cases} r, & \text{end} \\ r + \gamma \max Q(s', a; \theta^-), & \text{else} \end{cases}$$

$$q_{next} = Q(s, a; \theta)$$

$$Loss = (q_{target} - q_{next})^2;$$

 对于主网络参数 θ 使用 $Loss$ 进行梯度下降法更新网络参数 θ ;

 每隔 X 步更新一次 TD 网络, $\theta^- \leftarrow \theta$;

end

end

Output: 最优网络参数 θ

2.4 改进的 DQN 算法

由式2-6与式2-7可知, 不论是 Q-learning 还是 DQN, 值函数的更新公式中均有最大化操作, 通过最大化值函数网络的操作来选择行为。通过最大化值函数网络的操作来选择行为并进行评估, 整体上使得估计的值函数比真实的值函数大, 并且误差会随着动作空间的增加而增加。此时产生的过估计量往往是非均匀的, 故此时值函数的过估计就会影响到最优决策, 导致最终选择一个次优的动作。

为了解决 DQN 的不足，Double DQN 和 Dueling DQN 分别从网络的更新策略和网络的结构上做出了改进。

2.4.1 Double DQN 算法

为了解决值函数过估计的问题，Hasselt 提出了 Double DQN 方法^[16]。传统 DQN 中，选择行为指的是选择一个动作 a ，使其满足 $a = \arg \max_a Q(s', a; \theta^-)$ ；评估行为指的是利用 a 构建 TD 目标 q_{target} ， $q_{target} = r + \gamma \max Q(s', a; \theta^-)$ ，选择行为和评估行为用的是同一个 Q 网络及其网络参数。

Double DQN 分别采用不同的值函数来实现动作选择和动作评估。针对于传统的 DQN，由于传统 DQN 已经存在两个网络（主网络和 TD 网络），因此不需要改变 DQN 的网络结构，只需要改变 DQN 的参数更新策略。Double DQN 和 DQN 的区别在于：

(1) 首先使用主网络选择动作。

$$a = \arg \max_a Q(s', a; \theta)$$

(2) 其次使用 TD 网络找到该动作对应的 Q 值，构成 TD 目标。

$$\begin{aligned} q_{target} &= r + \gamma Q(s', a; \theta^-) \\ &= r + \gamma Q(s', \arg \max_a Q(s', a; \theta); \theta^-) \end{aligned}$$

此时构成的 q_{target} 在 TD 网络中不一定是最大的，但是该值是通过每步更新的主网络选取的最优动作，可以在一定程度上避免选到被高估的次优行为。Double DQN 的其余流程均与 DQN 相似，故算法流程不再赘述。

2.4.2 Dueling DQN 算法

在许多基于视觉感知的深度强化学习的任务中，不同的状态对应的值函数 $Q(s, a)$ 是不同的，但是在某些状态下，值函数的大小与动作无关。Baird 在 1993 年提出将 Q 值分解为价值（Value）和优势（Advantage）^[24]，即 $Q(s, a) = V(s) + A(s, a)$ 。 $V(s)$ 是在 s 状态下所有行为值函数（Q 值）关于行为概率的期望，即所有可能行为对应的 Q 值乘以该行为所对应的概率之和。 $A(s, a) = Q(s, a) - V(s)$ ，表示行为值函数相比于当前状态值函数的优势，即在这个状态下各个动作的优劣程度。

基于 Baird 的思想，将 DQN 用于竞争网络，就有了 Dueling DQN 算法的原理，

图2-3清晰地给出了 Dueling DQN 和 DQN 的网络结构差异。

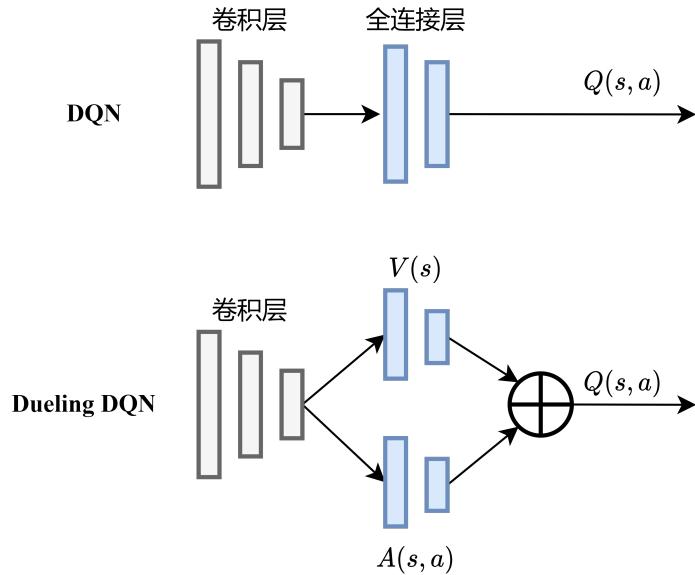


图 2-3 DuelingDQN 网络结构

Dueling DQN 将卷积层提取的抽象特征分流到两个支路上，两个支路所采用的全连接层结构相同。两个支路分别输出状态值函数 $V(s)$ 和优势值函数 $A(s, a)$ ，聚合函数将两个支路合并为 Q 函数：

$$Q(s, a; \alpha, \beta) = V(s; \beta) + A(s, a; \alpha)$$

但此式存在一个无法识别的问题，在确定的 Q 下，V 和 A 有无数可能，故需要对 A 值做出限定，强制令所有选择贪婪动作的优势函数为 0 (认为没有比选择贪婪动作的方法更优的选项了)。添加约束条件，此时 Dueling DQN 的行为值函数为：

$$Q(s, a; \alpha, \beta) = V(s; \beta) + A(s, a; \alpha) - \max_{a' \in A} A(s, a'; \alpha) \quad (2-8)$$

在实际中，一般使用优势函数的平均值代替上述最优值，虽然平均值改变了优势函数的值，但它可以保证缩小 Q 值的范围，去除多余的自由度，从而提高算法的稳定性。

$$Q(s, a; \alpha, \beta) = V(s; \beta) + A(s, a; \alpha) - \frac{1}{|A|} \sum_{a' \in A} A(s, a'; \alpha) \quad (2-9)$$

由于 Dueling DQN 与传统 DQN 有相同的输入输出，只需要改变网络结构和前向传播的公式，除此之外算法的处理流程和 DQN 是相同的。

2.5 本章小结

本章介绍了强化学习算法的组成及其马尔科夫决策模型的求解，通过对基于值函数（Value Based）的 Q-learning 算法及 DQN 算法的详细分析，由于自动驾驶决策控制任务无法进行精确的“查表”预知，需要进行神经网络的拟合，故采取基于 DQN 及其改进算法 Double DQN 和 Dueling DQN 算法完成自动驾驶决策控制任务。同时，根据输入数据的抽象程度，下一章节将对三种 DQN 算法进行决策与控制方面的设计与探究，从而得到 DQN 及其改进算法的对比实验效果。

第3章 基于DQN算法的自动驾驶避障算法设计

在第2章的分析中，DQN、Double DQN和Dueling DQN能够完成自动驾驶决策控制任务。本章将结合Highway-Env与Metadrive两种仿真环境，设计决策器与控制器，分别探究在抽象的数据输入与具体的数据输入的情况下，DQN及其改进算法对自动驾驶车辆决策与控制任务的效果和不同算法作用下的对比实验，从而满足功能需求。

3.1 基于DQN算法的决策器设计

3.1.1 仿真环境

Highway-Env是一个用于自动驾驶决策的最小仿真环境^[25]，它包含了自动驾驶决策任务的环境集合。对强化学习而言，Highway-Env包含了强化学习所需要的状态(state)、动作(action)和奖惩值(reward)设计。Highway-Env的仿真效果如图3-1所示，在这项任务中，自我车辆(ego-vehicle)正在一条多车道高速公路上行驶，该高速公路上存在其他车辆。智能体的目标是在避免与邻近车辆发生碰撞的同时达到高速，将车辆保持在最下方的道路行驶也会获得奖励。

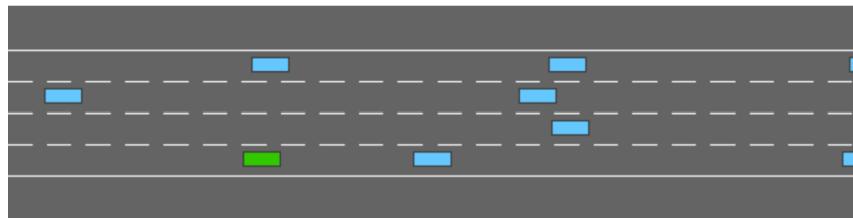


图3-1 highway-env仿真界面

本文采用Highway-Env进行自动驾驶决策任务的设计，智能体（自我车辆）根据周围车辆的环境情况，获取环境反馈的奖惩值函数，通过执行决策行为间接控制车辆避障。因此，首先需要明确Highway-Env的状态、动作和奖励函数设计。

Highway-Env的状态值设计

在Highway-Env中有一系列的数据状态集合，针对自动驾驶避障问题，应当获取Kinematics的状态函数。Kinematics的状态是一张 $V \times F$ 的表格，如表3-1所示，

V 表示用户定义的车辆数量， F 表示每个车辆的特征。第一列的 *Vehicle* 参数表示在当前画面中是否出现该车辆，如出现该车辆，则 *Vehicle* 值为 1，否则为 0。剩余的数值是车辆的坐标和速度，由于考虑的是自我车辆（ego-vehicle）的避障，所以环境中的其他车辆的坐标和速度均是相对于自我车辆的数值。

表 3-1 Kinetmatics 状态表格

Vehicle	x	y	v_x	v_y
ego-vehicle	x_{ego}	y_{ego}	v_{xego}	v_{yego}
vehicle 1	x_1	y_1	v_{x1}	v_{y1}
vehicle 2	x_2	y_2	v_{x2}	v_{y2}
...	...			
vehicle n	x_n	y_n	v_{xn}	v_{yn}

Highway-Env 的动作值设计

DQN 算法输入的是连续的状态，输出的是离散的动作。由于考虑的是决策问题，可以在决策之后增加一层低级的控制器，此处采用速度和转向控制器，使用 P 和 PD 控制器进行纵向和横向的解耦控制，使自我车辆能够以所需要的速度紧密跟随目标。关于低级控制器的部分在此不再赘述，表3-2表示了 DQN 的离散输出的决策所对应的动作量。动作值空间 A 的大小为 5。

表 3-2 决策状态表格

index	0	1	2	3	4
Action	左转	保持	右转	加速	减速

Highway-Env 的奖励函数设计

选择合适的奖励函数决定了网络参数优化的方向，在 Highway-Env 的简单的仿真环境中，不需要在奖励函数中具体指定预期驾驶行为的每一个方面（例如行车的速度和与前车的安全距离），因为奖励函数不能由某个模型来确定。直接指定一个较为简单和直接的函数便能在学习中看到效果。

在自动驾驶避障问题中，通常关注车辆的速度和避免碰撞，所以奖励函数由速度项和碰撞项组成，如式3-1所示， v, v_{min}, v_{max} 分别代表自我车辆的当前速度、最小速度和最大速度， a, b 是速度项和碰撞项的两个系数。通过调整 a, b 两个系数能够使得智能体的优化方向更加偏向高速或偏向安全性。

$$R(s, a) = a \frac{v - v_{min}}{v_{max} - v_{min}} - b \text{ collision} \quad (3-1)$$

3.1.2 网络结构设计

针对 Highway-Env 的状态值、动作值和奖励函数设计，由于 Highway-Env 的决策环境较简单，网络结构设计两层全连接神经网络作为拟合行为值（Q 值）函数的神经网络。如图3-2所示，此过程为决策器网络的前向传播示意图，自我车辆与其他车辆的状态由3.1.1节的状态值获得，为了匹配 DQN 网络全连接层的输入要求，将表格中 $V \times F$ 的二维数组压缩为一维数据，得到 $1 \times (V \times F)$ 的数据输入两层全连接神经网络。两层全神经网络之间使用 *relu* 函数激活，最终输出动作空间为 5 的决策动作。

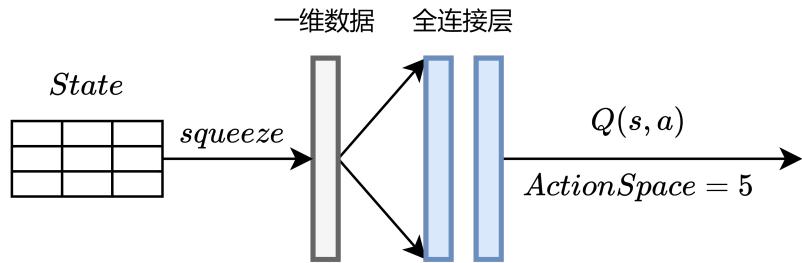


图 3-2 基于 DQN 算法的决策器网络

全连接层每一个结点都与上一层的所有结点相连，作用是将全连接层之前提取到的特征综合起来，最终输出行为值函数，以便后续环节的决策输出。根据前文 DQN 网络的相关研究，由于 DQN 网络需要实时的同智能体、环境交互，所以 DQN 网络一般采用较为简单的网络结构进行训练，其核心不在于网络结构的复杂程度，而在于算法的参数更新部分。和2.3节所示的 DQN 算法网络结构相比，Highway-Env 获取的状态值是周围车辆相对于自我车辆的位置与速度，每个数据具有明确的指向性，所以不需要通过卷积层提取整体数据的特征，只采用两层的全连接层网络结构能够

满足实验算法的准确度，又能保证运算速度和自动驾驶车辆决策动作的连贯。

3.1.3 决策器的学习过程

在决策器的网络结构设计结束后，决策器的算法需要根据2.3节所示的 DQN 算法进行构建。自动驾驶车辆首先获取当前的状态，每经过一个时间步，DQN 算法都会对自动驾驶车辆的动作和网络结构参数进行更新。对于自动驾驶决策器而言，决策器的学习过程主要分为前向动作选取和反向数据更新两个部分。

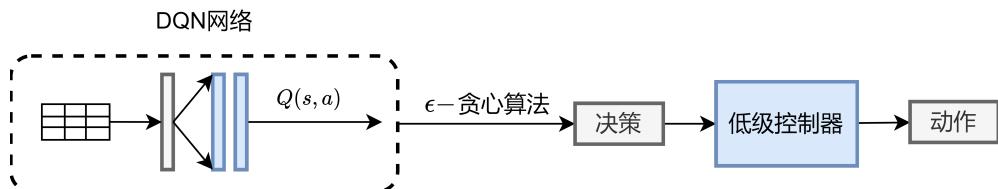


图 3-3 决策器前向动作选取

前向动作选取指的是在当前时刻，自动驾驶车辆将车辆当前的状态输入 DQN 的网络结构中，输出得到当前时刻车辆的决策动作。如图3-3所示，DQN 决策器网络输入自动驾驶车辆的当前状态，输出 1×5 的行为值函数 (Q 值)，在决策器学习过程中，采取 ϵ - 贪心算法将行为值函数对应为选取的策略。每个时间步中，针对 1×5 的行为值函数 (Q 值)，以 ϵ 概率随机选择任一随机动作，以 $1 - \epsilon$ 概率选择行为值函数最大的动作。 ϵ 的值随着训练次数的增加而逐渐减小，在一定的训练次数后最终使得智能体选择行为值函数最大的决策动作。得到了行为值最大的决策动作，通过低级的控制器对应到 Highway-Env 的动作空间，从而使自动驾驶车辆完成对应的动作。

反向数据更新指的是 DQN 网络根据经验回放库中的数据，对 DQN 网络参数进行的更新。根据2.3节所示的 DQN 算法，DQN 网络需要构建主网络 (Q_{next}) 和 TD 网络 (Q_{target})，并且需要通过两者差值的平方作为损失函数进行反向传递。从经验回放库中随机采样得到数据，对每组数据的执行如下操作：

$$\begin{aligned}
 q_{next} &= Q_{next}(s) \\
 q_{target} &= R(s, a) + \gamma \arg \max Q_{target}(s') \\
 Loss &= MSELoss(q_{next}, q_{target})
 \end{aligned} \tag{3-2}$$

这里的更新方式借鉴了 Q-learning 的更新方式，最终的网络结构和参数更新通过

Pytorch 进行构建。

经过以上对 Highway-Env 仿真环境的状态值、动作值、奖励函数的说明和对 DQN 决策器详细的网络结构设计与分析，下一章节将以上述理论和具体实现方法作为依据，针对 DQN 及其改进算法在 Highway-Env 仿真环境中进行自动驾驶决策任务的实验验证。

3.2 基于 DQN 算法的控制器设计

3.1节基于 DQN 算法分析了自动驾驶任务决策器的设计思路，对于 Highway-Env 的仿真环境，输入车辆坐标等一系列抽象数据，输出运动控制的决策信息。基于端到端的方法要求直接从感知输入产生动作，当包含传感器信息输入的车辆状态维度进一步增加，直接对自动驾驶车辆的控制量（转矩与轮转角）进行离散后输出，不经过低级控制器对车辆进行实时控制，是基于 DQN 算法的控制器希望达成的目标。区别于 Highway-Env 的最小仿真环境，基于 DQN 算法的控制器设计需要对 DQN 网络的输入进行扩充，也需要更加真实的仿真环境与训练场景。

3.2.1 仿真环境

表 3-3 仿真模拟器对比

Simulator	车辆动力学	多智能体	雷达、相机	真实数据导入	轻量级
CARLA ^[26]	✓	✓	✓	✓	
Metadrive ^[27]	✓	✓	✓	✓	✓
Highway-Env ^[25]					✓

MetaDrive 是一款高效的单车自动驾驶仿真模拟器^[27]，和 CARLA 模拟器^[26]类似，Metadrive 的状态、动作和奖励函数封装的较为完善，同时提供了较为准确的物理模型和多种传感器输入，优点是更加轻量级。多种仿真模拟器的对比如表3-3所示：

Metadrive 的仿真效果如图3-4，相较于 Highway-Env，它提供了更加具体的传感器输入和更加高维的观察输入，对于低级传感器（如 RGB 摄像头、深度摄像头和激光雷达）可以放置在场景中的任何位置，参数可调节，同时，Metadrive 还可以提供包括道路信息和附近车辆的速度和航向等高级场景信息作为状态量。本文采用 Metadrive 进行自动驾驶控制任务的设计。



图 3-4 metadrive 仿真界面

Metadrive 的状态值设计

Metadrive 提供一个状态向量，其中包含车辆导航任务所需的信息。状态向量由三部分组成：

(1) 车身状态

车身状态包括车辆当前的速度、转角、航向、控制量、横摆角速度和到边线的相对距离，这一部分包含了 9 个数据。

(2) 导航信息

导航信息包含了引导车辆驶向目的地的信息，Metadrive 首先计算车辆当前位置到目的地的路线，然后每组检查点以一定的间隔分散在整个路线上。到下一个检查点和下下一个检查点的相对距离和方向将作为导航信息给出，这一部分包含了 10 个数据。

(3) 传感器信息

传感器信息设置为灰度相机的图像输入，周围信息通过 80×80 的图像获取。针对于 DQN 算法，采用 4 帧 80×80 的灰度图作为输入，这一部分包含了 $4 \times 80 \times 80$ 个数据。

以上三部分数据在输入时都进行了归一化处理，都以 $[0, 1]$ 为区间进行了转换。

Metadrive 的动作值设计

Metadrive 接收标准化的动作作为输入进行车辆控制：

$$action = [steering, throttle]^T \in [-1, 1]$$

在每个时间步长中，Metadrive 将标准化的动作输入转换为轮转角 u_s 、发动机力 u_a 和制动力 u_b 。

$$u_s = S_{max}steering$$

$$u_a = F_{max}\max(0, throttle)$$

$$u_b = -B_{max}\min(0, throttle)$$

其中 S_{max} 是最大转向角， F_{max} 是最大发动机力， B_{max} 是最大制动力，通过这样的设计，每个智能体的动作空间总是固定的，这和标准化状态值设计是匹配的。

Metadrive 的奖励函数设计

由于 Metadrive 所面对的环境更加复杂，故 Metadrive 的奖励函数也包含更多的部分，完整的奖励函数由以下三部分组成：

$$R(s, a) = c_1 R_{driving} + c_2 R_{speed} + R_{termination} \quad (3-3)$$

(1) 驾驶奖励

$$R_{driving} = (d_t - d_{t-1}) \times lateral_factor$$

$$lateral_factor = 1 - \frac{2 \times lateral_now}{lane_width}$$

其中 d_t 和 d_{t-1} 表示目标车辆在当前车道上连续两个时间步长的纵坐标，该奖励将鼓励智能体向前移动。 $lateral_factor$ 表示车辆是否远离了当前车道的中心，该奖励将鼓励智能体进行车道保持。

(2) 速度奖励

$R_{speed} = v_t/v_{max}$, v_t 和 v_{max} 表示当前速度与最大速度。该奖励将鼓励智能体尽可能接近 v_{max} 。

(3) 终止奖励

$R_{termination}$ 是一组离散的奖惩函数，其包含了几种智能体可能的情况：

state	车辆到达目的地	车辆驶出道路	车辆与其他车辆相撞	车辆撞到障碍物
cost	+10.0	-5.0	-5.0	-5.0

3.2.2 网络结构设计

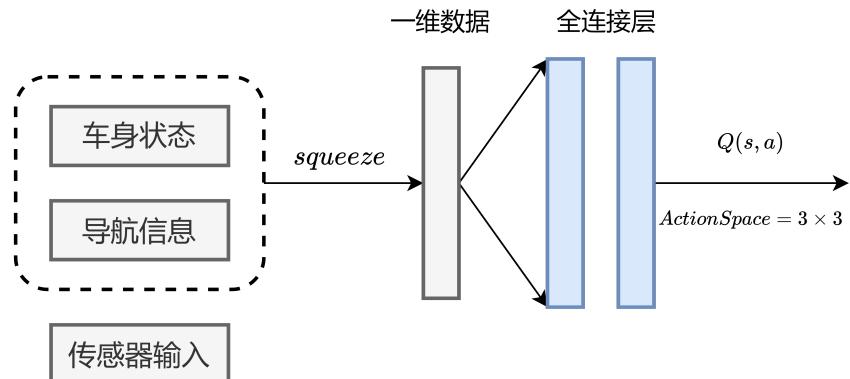


图 3-5 基于 DQN 算法的控制器网络

Metadrive 的状态量包括车身状态、导航信息和传感器信息，针对每个时间步长，车身状态和导航信息共包含 19 个数据，传感器信息包含 $4 \times 80 \times 80 = 25600$ 个数据，因此前者与后者需要做出区分。网络结构设计需要明确网络结构与动作空间的大小，出于验证用途的考量，DQN 的控制器设计预期实现使车辆在预定轨道上的循迹功能。参照 Highway-Env 的网络结构设计，仍采用两层全连接神经网络作为神经网络的结构，全连接层之间采用 *relu* 函数进行激活。在传感器信息方面，传感器获取周围车辆与道路信息，但是其获取的道路信息在车身状态与导航信息中已有体现，针对于车辆在预定轨道上的循迹功能，仅根据车身状态和导航信息便能依据全连接网络的综合能力完成相关任务，所以此时不将传感器信息作为网络结构的输入。具体的网络结构如图3-5所示。

在动作空间的设计方面，动作空间分为 $[steering, throttle]$ 两个分量，针对自动驾驶车辆的控制器而言，控制器要求在 $50Hz - 100Hz$ 的频率下做出响应，由于每个时间步长的间隔较短，且考虑到算力的限制，将连续的动作空间分为 3×3 的均匀

表 3-4 控制器动作空间

steering	-0.5	0	1
index	0	1	2
throttle	-0.5	0	0.5
index	0	1	2

区间并对其进行编号，令 $action_index = 3 \times steering_index + throttle_index$ ，通过输出的索引确定输出的动作值。

3.2.3 控制器的学习过程

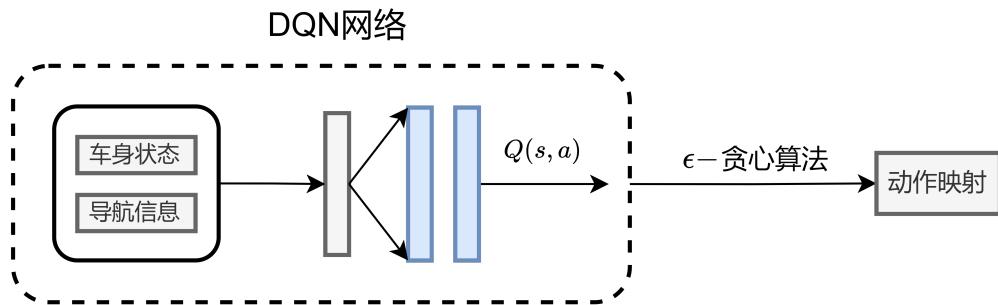


图 3-6 控制器前向动作选取

区别于 Highway-Env 的决策动作输出，Metadrive 使用控制量进行端到端的直接输出，如图3-6所示，DQN 控制器网络输入自动驾驶车辆的车身状态与导航信息，输出 3×3 的行为值函数 (Q 值)，在控制器学习过程中，采取 ϵ - 贪心算法将行为值函数对应为 $action_index$ ，并根据表3-4映射到 $[steering, throttle]$ 的实际控制量。尽管和决策器相比输入的信息含义并不相同，但通过全连接层的综合能力，能够将提取到的特征值进行综合，从而得到较为稳定的输出结果。

反向数据更新仍然是构建主网络 (Q_{next}) 和 TD 网络 (Q_{target})，并通过两者差值的平方作为损失函数进行反向传递。从经验回放库中随机采样得到数据，对每组数据的执行如下操作：

$$\begin{aligned}
 q_{next} &= Q_{next}(s) \\
 q_{target} &= R(s, a) + \gamma \arg \max Q_{target}(s') \\
 Loss &= MSELoss(q_{next}, q_{target})
 \end{aligned} \tag{3-4}$$

这里的更新方式借鉴了 Q-learning 的更新方式，最终的网络结构和参数更新通过 Pytorch 进行构建。

3.3 本章小结

本章介绍了基于 DQN 的自动驾驶避障算法设计，将 DQN 算法及其改进算法应用于自动驾驶决策器与控制器中，主要分析了自动驾驶决策器和控制器应用的仿真环境、状态值、动作值和奖励函数的设计，在第 2 章的算法研究的基础上，对自动驾驶决策器和控制器的网络结构设计和学习过程做了详细的分析与设计。至此，已经将基于 DQN 算法的决策器设计和基于 DQN 算法的控制器设计的理论和具体实现方法进行了详细的说明，后文将进行具体的实验对设计的算法进行验证。

第 4 章 仿真实验验证

第 2 章和第 3 章对基于 DQN 及其改进算法的决策器、控制器的算法原理、网络结构和具体实现进行了分析，本章将对基于改进型 DQN 的决策器和控制器进行实验验证。基于 DQN 算法的决策器使用 Highway-Env 作为实验仿真环境；基于 DQN 算法的控制器使用 Metadrive 作为实验仿真环境。决策器和控制器实验均对比 DQN 及其改进算法的影响，主要验证在两种仿真环境中，经过多次训练的 DQN 算法的收敛性和可行性，同时对比 DQN 算法的改进型 Double DQN 和 Dueling DQN，对比每个 episode 中的行为值函数、总体奖励、平均奖励和损失函数（Loss）的差异，从而根据结果对 DQN 及其改进算法进行效果改进和优劣分析。

4.1 基于 DQN 算法的决策器实验

基于 DQN 算法的决策器使用 Highway-Env 作为实验仿真环境，搭建第 3 章所述的决策器神经网络，通过 Pytorch 的 API 对自动驾驶决策器进行实现。具体实现的代码已上传至 Github 中以便查阅^[28]。

实验参数

表 4-1 决策器网络参数

参数	参数值
学习率 α	1e-3
折扣因子 γ	0.9
贪心算法参数 ϵ	0.9
TD 网络更新步数 N	1000
经验回放库 D	2000
经验回放库抽取批次 $batch$	32
训练回合数 $episode$	24000

由于 Highway-Env 的实验环境较简单，对于 DQN 网络的经验回放，不需要容量太多的经验回放库便能学习得到目标，实验参数如表4-1。同时，抽取的经验回放库批次更新的 $batch_size$ 采用 32 作为参数，兼顾了训练的精度和准确性。对于训练回

合数 *episode*, 由于目的是探究 DQN 及其改进算法的收敛性和参数趋势, 故采取较长时间的回合数进行训练。

仿真效果

在 Highway-Env 下, 基于 DQN 算法的决策器仿真效果如图4-1所示, (a) ~ (f) 代表仿真车辆完成自动驾驶避障的过程, 仿真实验采用的方法是在一段定长的时段, 随机生成车辆周围的障碍 (此处是其他车辆) 信息, 一旦自我车辆撞击障碍则仿真环境结束并输出最终奖励。 (a) ~ (f) 的过程是在仿真过程中截取的一段典型的避障环节, 可以看到, 自我车辆能够自主地进行周围障碍的识别并躲避。

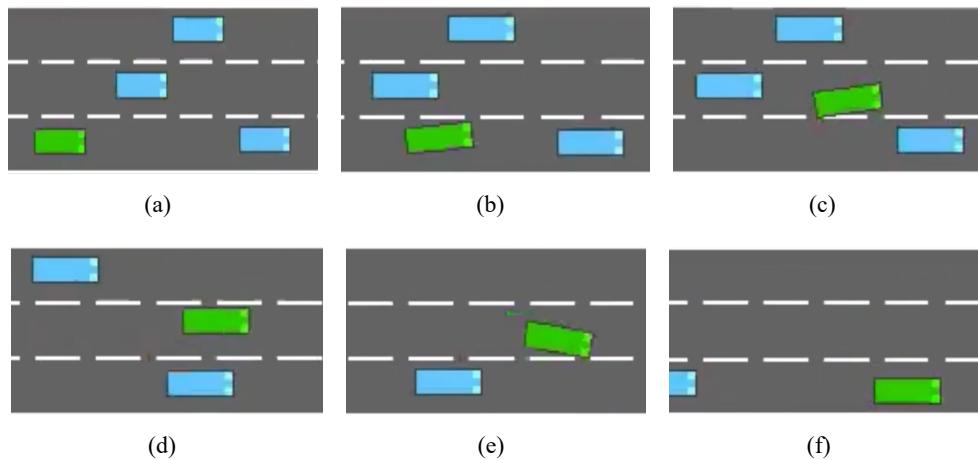


图 4-1 基于 DQN 算法的决策器仿真效果

以上行驶过程是智能体经过 10000 次训练达成的结果, 经过 10000 次的训练, 自动驾驶车辆基本学会了主动换道与避障的策略, 能够使自动驾驶车辆在一定的时间段内稳定行驶到终点。下面将通过具体的数据对 DQN 算法的输出进行分析。

基于 DQN 算法的决策器收敛性

在验证 DQN 算法的效果前, 需要判定 DQN 算法的收敛性。图4-2(b)表示的是整个训练过程中损失函数 Loss 的变化, 此处以参数每更新 10 万次进行采样, 损失函数 Loss 的值在 0.02 以下, 而图4-2(a)显示的是 DQN 算法的 Q 值, 在 2 万次的训练次数后总体趋近于 8.5 的区间内。

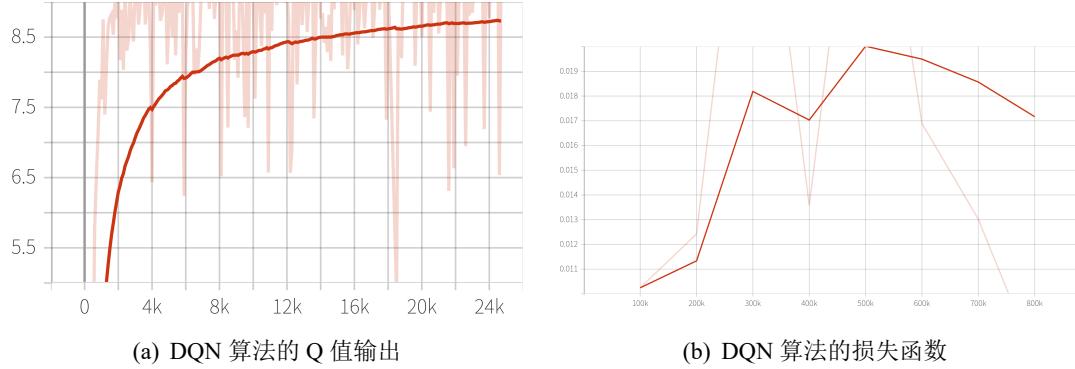


图 4-2 DQN 算法的收敛性

$$error = \frac{Loss}{Q_Value} = \frac{0.02}{8.5} < 0.5\% \quad (4-1)$$

根据图4-2所示，由于误差 $< 1\%$ ，此时可以判定 DQN 算法的收敛。

基于 DQN 及其改进算法的决策器对比实验

本实验验证了基于 DQN、Double DQN 和 Dueling DQN 算法的自动驾驶决策器，图4-3表示了三种算法的收敛性对比，其中红、蓝、橙三色曲线分别代表 DQN、Double DQN、Dueling DQN 的输出收敛情况。

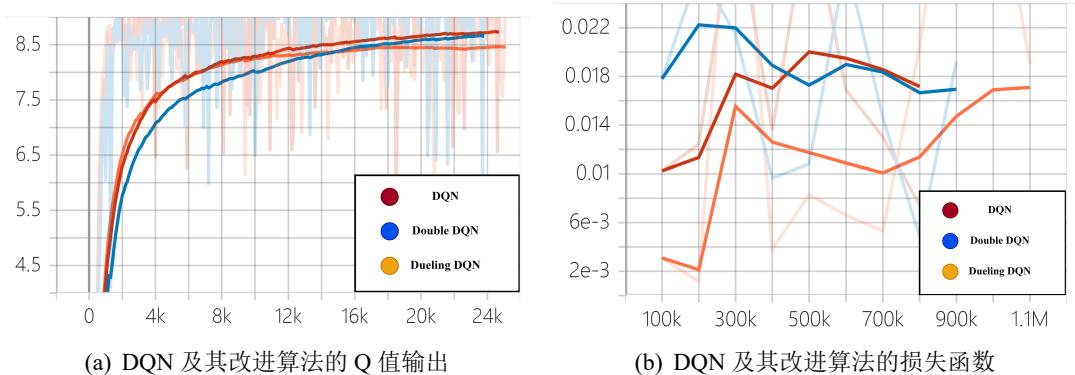


图 4-3 DQN 算法及其改进算法的收敛性对比

根据式4.1所示，由于 Double DQN 和 Dueling DQN 输出的 Q 值与损失函数与

DQN 差异不大，故三种算法在实验条件下均收敛。针对2.4节提出的 DQN 存在的问题，由图4-3(a)能够看出，Double DQN 算法解决了 DQN 算法的值函数过估计的问题，Dueling DQN 算法相较于 DQN 算法也更早的进入了收敛状态。

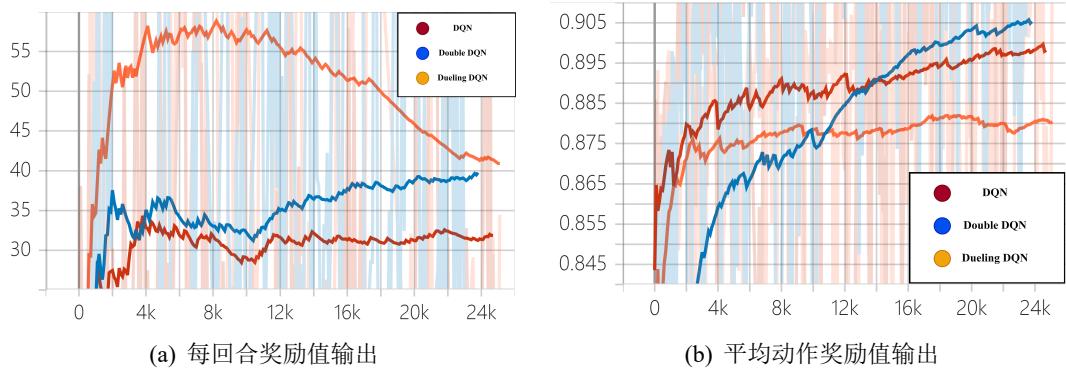


图 4-4 DQN 算法及其改进算法的奖励值对比

由图4-4(a)可知，Dueling DQN 每回合得到的奖励值输出最高，在 1 万 2 千次训练后开始逐步下降至与 Double DQN 与 DQN 的奖励值输出接近，这是由 DQN 网络的实验参数（也称为超参数）和3.1.1节所表示的奖励函数式3-1共同决定的，Dueling DQN 在初始阶段学习到的策略在某个状态下能够得到较高的奖励函数，随着训练次数的提升和获取状态量的增加，Dueling DQN 学习到了更加稳健的策略，奖励函数由原来的追求高速转向安全性的要求。

由图4-4(b)可知，Double DQN 的平均动作奖励值输出在 1 万 2 千次训练后开始逐步上升，Double DQN 分别采用不同的值函数来实现动作选择和动作评估，相对于最大化操作的值函数更新具有更加明显优势。

通过对以上仿真实验结果和数据的分析，DQN 及其改进算法在 Highway-Env 仿真实验环境中完成自动驾驶决策任务能够得到收敛，对于平均动作奖励，Double DQN 因其在动作选择和动作评估上的参数更新改进，能够得到相比于 DQN 和 Dueling DQN 更加良好的决策策略输出。

4.2 基于 DQN 算法的控制器实验

在基于 DQN 算法的决策器实验过程中，Double DQN 算法对于平均动作奖励值输出有相对优异的表现，本节将 DQN 及其改进算法直接作用于动作输出，在不改变网络结构的情况下，重点关注 Double DQN 和其他 DQN 算法的对比实验效果。

基于 DQN 算法的控制器器使用 Metadrive 作为实验仿真环境，搭建第 3 章所述的控制器神经网络，通过 Pytorch 的 API 对自动驾驶控制器进行实现。具体实现的代码已上传至 Github 中以便查阅^[29]。

实验参数

表 4-2 控制器网络参数

参数	参数值
学习率 α	1e-3
折扣因子 γ	0.9
贪心算法参数 ϵ	0.9
TD 网络更新步数 N	1000
经验回放库 D	50000
经验回放库抽取批次 $batch$	32
训练回合数 $episode$	100000

由于 Metadrive 的状态输入涉及车身状态和导航信息等多样信息，对于 DQN 网络的经验回放，需要更多的经验回放库学习得到预期目标，实验参数如表4-2，经验回放库 D 容量扩大为 50000，通过扩大经验回放库的方式使得智能体获取到更多的状态，避免智能体长时间获取单一状态集，失去算法的泛化性。对于训练回合数 $episode$ ，为了得到稳定的 DQN 及其改进算法的参数趋势和应用性能，采取较长回合数进行训练，在实际的仿真实验环境中，经过约 20000 次的训练，DQN 及其改进算法均已收敛，但得到稳定的参数仍需更多回合的训练次数，故将训练回合数设定为 100000。

仿真效果

在 Metadrive 的仿真环境下，图4-5体现的是基于 Double DQN 算法的控制器仿真效果，(a) ~ (f) 代表仿真车辆完成自动驾驶路线循迹的过程，仿真实验采用的方法是在一段定长的时段，随机生成地图信息，一旦自我车辆触及黄线或逆向行驶，则仿真环境结束并输出最终奖励。(a) ~ (f) 的过程是在仿真过程中截取的一段典型的循迹和换道环节，针对环形道路和直道等多样环境，自我车辆能够自主地进行周围信息的识别并较为稳定地输出轮转角和转矩 [steering, throttle]，根据画面中的导航点信息，自主实现换道和车道保持循迹功能。

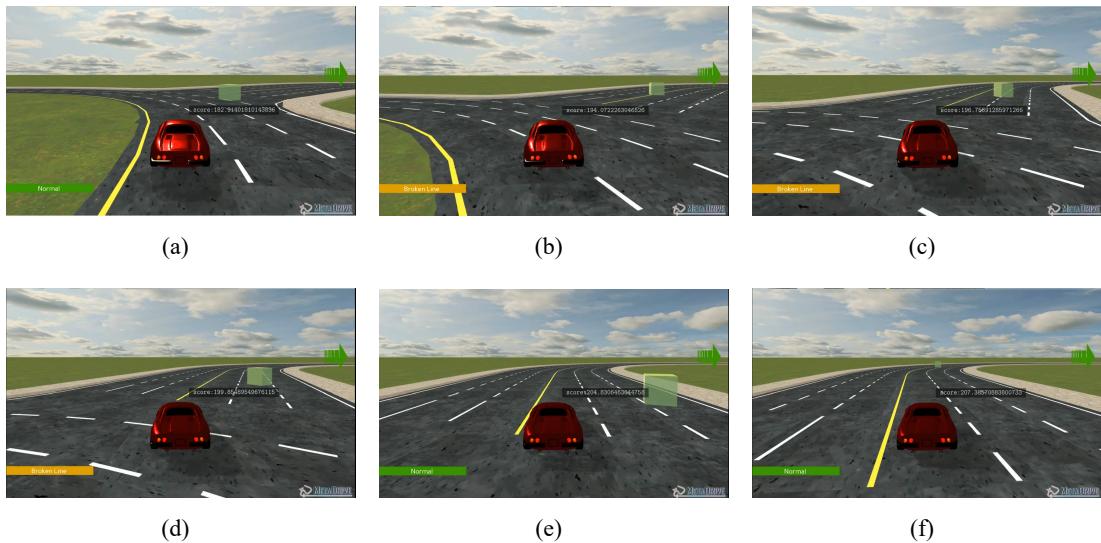


图 4-5 基于 DQN 算法的控制器仿真效果

以上行驶过程是智能体经过 20000 次训练达成的结果，经过 20000 次的训练，自动驾驶车辆基本学会了主动换道与循迹的策略，能够使自动驾驶车辆在一定的时间段内稳定行驶到终点。根据4.1节的结论，不再单独对 DQN 的输出结果进行分析，主要分析 Double DQN 和其他 DQN 算法的对比实验效果。

基于 DQN 及其改进算法的控制器对比实验

本实验验证了基于 DQN、Double DQN 和 Dueling DQN 算法的自动驾驶控制器，图4-6(a)和图4-6(b)表示了三种算法的收敛性对比，其中红、蓝、橙三色曲线分别代表 DQN、Double DQN、Dueling DQN 的输出收敛情况。

对于 DQN 和 Double DQN:

$$error = \frac{Loss}{Q_Value} = \frac{0.3}{8.5} < 4\% \quad (4-2)$$

对于 Dueling DQN:

$$error = \frac{Loss}{Q_Value} = \frac{0.3}{3.5} < 10\% \quad (4-3)$$

由于 Dueling DQN 输出的 Q 值相较于 Double DQN 和 DQN 算法过小，前者误差相对于后者较大，Dueling DQN 的收敛程度较低，所以本节重点探究 DQN 和 Double DQN 的对比实验结果。

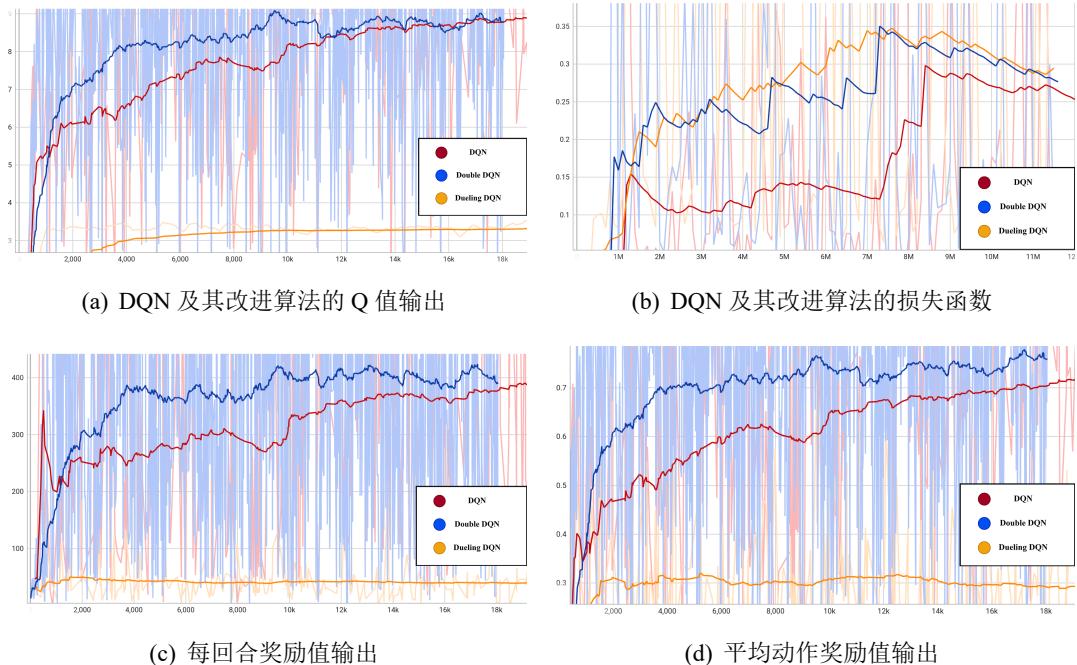


图 4-6 DQN 算法及其改进算法的控制器对比实验

由图4-6(c)和图4-6(d)可知，Double DQN 每回合得到的奖励值输出最高，平均动作奖励值也最高，这印证了4.1节中对 Double DQN 算法在动作选择和动作评估方面的分析，在更加复杂的环境中，Double DQN 采取的网络参数更新策略避免选到被高

估的次优行为，相对于最大化操作的值函数更新具有更加明显优势。

在4.1节中每回合奖励值输出函数最高的 Dueling DQN 算法在存在更加复杂输入的 Metadrive 仿真环境中表现不佳，一方面的原因是其两个支路所采用的全连接层结构较为简单，无法综合状态值函数 $V(s)$ 和优势值函数 $A(s, a)$ 使其满足自动驾驶车道保持和循迹的任务要求；另一方面的原因是其采用最大化操作的网络参数更新策略，存在选到被高估的次优行为的概率。

通过对以上仿真实验结果和数据的分析，DQN 及其改进算法在 Metadrive 仿真实验环境中完成自动驾驶控制任务能够得到收敛，对于平均动作奖励和总体动作奖励，Double DQN 能够得到相比于 DQN 和 Dueling DQN 更加良好的控制行为输出。

4.3 本章小结

本章根据第 3 章基于 DQN 算法的自动驾驶避障算法设计，进行了基于 DQN 及其改进算法的自动驾驶决策器和控制器实验，验证了基于 DQN 及其改进算法在自动驾驶决策器和控制器上的可行性和有效性。

针对自动驾驶决策器和控制器，Dueling DQN 因其网络结构设计较为简单，在复杂性较低的自动驾驶决策仿真环境中表现较好，在复杂性较高的自动驾驶控制仿真环境中需要进一步改进网络结构。Double DQN 因其在动作选择和动作评估上的参数更新改进，在两种仿真环境中都能够收敛，并且得到了相对于 DQN 和 Dueling DQN 更加良好的决策策略和控制行为输出。

结 论

本文提出的方法实现了基于 DQN 及其改进算法的自动驾驶决策器和控制器，并就 DQN 及其改进算法开展了较为深入的研究。实验结果表明，本文设计实现的基于 DQN 及其改进算法的自动驾驶决策器和控制器达到了预期的决策和控制效果，能够有效提高自动驾驶车辆在决策和控制中的准确性和鲁棒性。对于较为简单的网络结构，DQN 特别是 Double DQN 算法由于其改进的动作选择和评估方法，能够获得更加稳定有效的行为策略。

DQN 算法作为无模型的强化学习算法，通过离散化动作空间，能够应用于连续状态空间。本文对于 DQN 算法及其多种改进算法在自动驾驶车辆决策控制方面的应用，不仅为对自动驾驶领域类似的复杂动力学模型问题的解决提供了新的思路，也为自动驾驶的决策与控制提供了借鉴和参考。

但是，本文对强化学习算法在自动驾驶的决策和控制方面的研究还处在初级阶段，目前研究的是自动驾驶决策器和控制器在连续状态空间和离散动作空间的情况，对于连续状态空间和连续动作空间（如 DDPG、PPO 等算法）还需要进行更加深入的研究。

对于复杂多样的的传感器输入，仅使用传统 DQN 直接由输入灰度图像输出动作，不添加车辆信息与导航信息，控制效果不佳。未来考虑采用卷积神经网络（如 ResNet^[30]）对传感器信息特征值进行提取，同时，由于传感器输入需要与车辆信息和导航信息组合，所以仍需要对 DQN 网络结构进行更加深入的研究和改动。

参考文献

- [1] 唐振韬, 邵坤, 赵冬斌, 等. 深度强化学习进展: 从 AlphaGo 到 AlphaGo Zero[J]. 控制理论与应用, 2017, 34(12): 18.
- [2] Li Y. Deep Reinforcement Learning: An Overview[J]. ArXiv preprint arXiv:1701.07274, 2017.
- [3] 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.
- [4] Yurtsever E, Lambert J, Carballo A, et al. A survey of autonomous driving: Common practices and emerging technologies[J]. IEEE access, 2020, 8: 58443-58469.
- [5] Chen C, Seff A, Kornhauser A, et al. Deepdriving: Learning affordance for direct perception in autonomous driving[C]//Proceedings of the IEEE international conference on computer vision. 2015: 2722-2730.
- [6] Pomerleau D A. Alvinn: An autonomous land vehicle in a neural network[J]. Advances in neural information processing systems, 1988, 1.
- [7] Bojarski M, Del Testa D, Dworakowski D, et al. End to end learning for self-driving cars[J]. ArXiv preprint arXiv:1604.07316, 2016.
- [8] Baluja S. Evolution of an artificial neural network based autonomous land vehicle controller[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 1996, 26(3): 450-463.
- [9] Sallab A E, Abdou M, Perot E, et al. Deep reinforcement learning framework for autonomous driving[J]. Electronic Imaging, 2017, 2017(19): 70-76.
- [10] Zhu Y, Mottaghi R, Kolve E, et al. Target-driven visual navigation in indoor scenes using deep reinforcement learning[C]//2017 IEEE international conference on robotics and automation (ICRA). 2017: 3357-3364.
- [11] Zhao D, Chen Y, Lv L. Deep reinforcement learning with visual attention for vehicle classification[J]. IEEE Transactions on Cognitive and Developmental Systems, 2016, 9(4): 356-367.
- [12] Chen D, Koltun V, Krähenbühl P. Learning to drive from a world on rails[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 15590-15599.
- [13] Ha D, Schmidhuber J. Recurrent world models facilitate policy evolution[J]. Advances in neural information processing systems, 2018, 31.
- [14] Lange S, Riedmiller M, Voigtlander A. Autonomous reinforcement learning on raw visual input data in a real world application[C]//The 2012 international joint conference on neural networks (IJCNN). 2012: 1-8.
- [15] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. ArXiv preprint arXiv:1312.5602, 2013.
- [16] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]//Proceedings of the AAAI conference on artificial intelligence: vol. 30: 1. 2016.
- [17] Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[C]//International conference on machine learning. 2016: 1995-2003.
- [18] Zong X, Xu G, Yu G, et al. Obstacle avoidance for self-driving vehicle with reinforcement learning[J]. SAE International Journal of Passenger Cars-Electronic and Electrical Systems, 2018, 11(1): 28-38.

北京理工大学本科生毕业设计（论文）

- [19] Gao F, Geng P, Guo J, et al. ApolloRL: a Reinforcement Learning Platform for Autonomous Driving[J]. ArXiv preprint arXiv:2201.12609, 2022.
- [20] Monahan G E. State of the art—a survey of partially observable Markov decision processes: theory, models, and algorithms[J]. Management science, 1982, 28(1): 1-16.
- [21] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. 1999: 229-235.
- [22] Watkins C J, Dayan P. Q-learning[J]. Machine learning, 1992, 8(3): 279-292.
- [23] Lin L J. Reinforcement learning for robots using neural networks[M]. Carnegie Mellon University, 1992.
- [24] Baird III L C. Advantage updating[R]. WRIGHT LAB WRIGHT-PATTERSON AFB OH, 1993.
- [25] Leurent E. An Environment for Autonomous Driving Decision-Making[Z]. <https://github.com/eleurent/highway-env>. 2018.
- [26] Dosovitskiy A, Ros G, Codevilla F, et al. CARLA: An Open Urban Driving Simulator[C]// Proceedings of the 1st Annual Conference on Robot Learning. 2017: 1-16.
- [27] Li Q, Peng Z, Xue Z, et al. MetaDrive: Composing Diverse Driving Scenarios for Generalizable Reinforcement Learning[J]. ArXiv preprint arXiv:2109.12674, 2021.
- [28] Huang C. Highway-Env Agent using DQN[Z]. <https://github.com/HReX39/Highway-Env-DQN>. 2022.
- [29] Huang C. Metadrive-simulator Agent using DQN[Z]. <https://github.com/HReX39/Metadrive-DQN>. 2022.
- [30] Targ S, Almeida D, Lyman K. Resnet in resnet: Generalizing residual architectures[J]. ArXiv preprint arXiv:1603.08029, 2016.