摘　要

本文提出了一个新的东南大学 LaTeX 硕士研究生毕业论文模板，并说明了如何更优雅地写出一篇漂亮而无用的文章。

关键词： TeX, LaTeX, 学位论文

Abstract

This article proposes a new Southeast University master degree thesis LaTeX template and explains how to elegantly write an article which is beautiful but full of shit.

Keywords: TeX, LaTeX, Thesis

目　录

第一章　绪论

1.1　研究工作的背景及意义

随着人口、就业和社会活动的增加，出行需求的增长往往是城市发展不可逆转的结果。这将导致一些大型城市地区的交通状况恶化，在高峰期时，一些主干道的通行能力将会低于出行的出行需求，出现拥堵现象。为了获得准确的出行需求预测并实施有效的需求管理策略，研究人员和政府机构了解出行者在出行时如何进行决策将会至关重要。一旦决策者知道出行者在何时何地以及将采取什么模式出行，就可以提供有效的解决方案来缓解拥堵。因此，出行决策建模成为交通研究的关键。

研究人员通常将出行决策描述为不同维度的备选方案选择，例如出发时间、目的地、方式和路线。这些选择问题通常被描述为离散或者连续选择模型。早期的出行决策模型只考虑了一个维度，即从该选择维度的一组相互排斥的备选方案中选择了一个备选方案。然而在实际生活中，需要结合不同行为维度的进行多维决策才足以支持日益增长的拥堵管理策略应用。

近年来，多维出行选择模型得到了更多的关注，因为与传统的一维选择模型不同，多维出行选择模型诠释了不同选择的相关性。在出行选择的问题中，模式选择和出行时间选择是两个非常重要的模块。从个人层面上看，这些都是出行者出行需求的重要性决策。在集计层面上，这决定了交通网络的荷载以及它的时空分布。不同交通方式的可行性与吸引性都取决于它的服务水平，如等待时间、出行时间、出行成本等，这可能会受到各种政策措施的影响，如高峰时段定价、拥堵定价、共乘或公交使用激励。对此类政策措施的评估需要一个出行模式和出发时间选择的综合框架。

模式选择与出行时间选择的研究大多基于随机效用最大化的离散选择模型。如嵌套 Logit 模型、交叉嵌套 Logit 模型和混合 Logit 模型可以应用于多维选择问题[1]，因为它们具有建模不同选择维度之间相关性的能力。利用随机效用最大化的模型依靠着其强大的理论依据而被广泛地应用。基于随机效用的模型是可以解释出行选择的基本理论，而对于复杂的决策过程建模的适用性，尤其是在选择预测中，可能会受到随机效用函数中线性结构的限制。对于多维选择问题，不同维度之间的关联结构也需要预先确定。

效用的随机成分不仅可以解释出行者对与观察信息的局限性，而且可以考虑决策者的不完全信息和偏好的随机变化。然而，以下事实支持了对基于学习方法的出行选择模型的需求。首先，乘客在模式选择的决策过程，是由不同出行方式的服务水平信息告知和指导的。这些知识通常是通过各种方式获得的 (包括出行经验)，并且会随着时间动态变化。第二，出行决策受到一些行为因素的影响，其中乘客更倾向于 (更少) 选择 (改变) 他们已经习惯的模式。第三，交通系统的随机性和时间依赖性最可能引起出行者的自适

应模式切换决策，在这种决策中，出行者可能会根据以往的经验更新他们对每种出行模式的预期效用。传统方法不能解决决策过程中涉及的时间维度。因此，与传统的选择建模方法相比，基于学习的出行决策模型更可取。

近年来, 强化学习因其强大的探索能力和自主学习能力, 已经与监督学习、无监督学习并称为三大机器学习技术 [2]. 伴随着深度学习的蓬勃发展, 功能强大的深度强化学习算法层出不穷, 已经广泛应用于游戏对抗、机器人控制、城市交通和商业活动等领域, 并取得了令人瞩目的成绩 [3]。后续大量的研究成果也表明, 强化学习是实现通用人工智能的关键步骤.

出行选择的决策过程是一个复杂的过程，会受到环境的影响而不断地变化，通过建立传统的出行选择模型来解释出行行为的方法过于理想化。而此类场景很好地契合了强化学习“无模型、自学习、数据驱动”，使用强化学习的方法可以将此类复杂的模型使用深度神经网络进行描述，通过提取不同外界环境的特征数据如等待时间、出行成本等构建状态输入，再对出行者的出行行为进行优化，利用大数据训练网络增加其真实性和可靠性。相较于传统的离散选择模型，强化学习的方法对复杂的场景适应能力有极大的提升，并且适用的场景更加广泛。

1.2 国内外研究

在出行选择的模型中，通常使用基于随机效用的离散选择模型对不同维度的选择行为进行建模。从 McFadden[2] 在 1973 年提出了著名的 Multinomial Logit（MNL）模型用于行为选择建模以来，Logit 系列模型就被广泛应用于出行决策问题。然而，MNL 存在一个被公认的问题：它假设了不相关的替代方案的独立性，也被称为 IIA（independence of irrelevant alternatives）特性。这表示替代方案未观察到的特征彼此之间相互独立，然而在一些出行选择的问题中这个假设将会不成立。例如，在离散出发时间选择中，相邻出发时间区间的未观测特征往往表现出显著的相关性。为了解决这一问题，Ben-Akiva[5] 等在 1998 年提出了 Nested Logit（NL）模型和有序广义极值模型（Ordered Generalized Extreme Values, OGEV）。NL 模型能够识别嵌套组内不同替代方案之间的相关性。有序广义极值模型允许为每一对分组的备选方案提供一个相关参数。经过 Bhat[6-8] 在 1998 年的测试，得出的结论是，NL 和 OGEV 模型的性能都优于 MNL。在此之后，不同的研究人员针对问题的多样性提出了更先进的 NL 模型，如 Ben-Akiva 和 Bierlaire [9] 在 1999 年提出的 cross-nested Logit（CNL）模型和 Lemp[10] 等在 2010 年提出的连续 CNL 模型。另一种改进的离散选择模型是 De Jong 等在 2003 年提出的 mixed Logit(MMNL) 模型[3]，它通过改变 MNL 模型的参数随给定分布变化来考虑个体之间的异质性。然而，MMNL 的一个限制是，它需要对整个人口的参数分布进行特定的假设。这种限制可以通过潜在类 (LC) 模型来解决，该模型可以通过将总体划分为离散数量的类来捕获未观察到的偏好异质性[4]。

另一种研究出行决策的主流方法是机器学习。与统计方法不同，在统计方法中，研

究人员试图确定模型结构和需要估计的参数，机器学习方法关注的是数据本身，并试图找到不同参数之间的关联。相较于随机效用的模型，机器学习模型的结构更加灵活，方便其探索不同特征之间的关联。针对出行决策的建模，主要有以下几种主流的机器学习方法：决策树模型 [11]，神经网络模型 [12]，以及支持向量机 [13]。与随机效用离散选择模型相比，这些机器学习方法可以处理大型数据库。然而，机器学习方法很少能捕捉到对出行行为研究较为重要的因素，包括时间价值 (VOT) 和弹性。此外，使用机器学习方法作为模型的主要框架还存在一个限制是机器学习模型对训练数据很敏感，在样本不足或有偏倚的情况下，会导致欠拟合或过拟合问题。

强化学习作为一种被广泛应用的学习机制, 是利用环境的反馈评价作为学习的输入, 学习主体拥有较强的环境适应能力的机器学习方法, 因此适用于重复日变的交通决策场景中。强化学习被用来解决各种领域的顺序决策问题，如机器人控制、电子游戏和系统优化等。强化学习的理论为人类行为提供了可解释的心理学和神经科学视角，即人类如何在给定的环境中计划自己的行为。此外，强化学习框架提供了智能决策的数学形式化形式，在智能体控制中具有强大而广泛的适用性，可直接应用于控制理论中顺序决策问题的求解。在交通领域，强化学习方法也受到了广泛的应用，例如交通流管理、自动驾驶，以及路线规划 [14]。近期，一些研究已经采用强化学习方法来建模出行者日常活动计划以及出行决策。

现有的出行决策与出行需求预测的研究工作多使用基于价值的强化学习方法，Janssens[15] 在 2007 年使用 Q-learning 的强化学习方法解决活动调度问题。Vanhulsel[16] 等在 2009 年通过基于 Q-learning 的方法构建 MATSim 结构方程模型。Medhat[17] 等在 2008 年开发了一个更全面的动态公共交通路径和出行活动选择模型，称为 MILITRAS 系统，其中的模型使用了预先设定的奖励 (效用) 函数。

近几年，深度强化学习在控制复杂智能体的决策行为上取得了巨大的成功，并将强化学习算法与许多神经相关因素的研究相结合，激发了大量使用人工神经网络作为通用函数逼近器的强化学习方法的研究。Hausknecht[18] 等在 2016 年发表的著作研究了使用深度强化学习方法与多智能体合作行为。值得注意的是，它将多智能体研究中的矩阵博弈推广到更复杂的状态和行动空间。

1.3 本文的贡献与创新

在使用强化学习的仿真环境中，可以根据不同出行场景将出行者主要分成两种：有电子地图导航和无电子地图导航。在有电子地图导航的场景中，出行者信赖电子地图导航，会根据导航信息一般选择行程最短的模式和路径行驶。在此场景中，出行模式选择问题将转变为备选路径的行程时间预测和预估价问题。可以考虑使用预计到达时间（ETA）的计算方法解决 [22]。在无电子地图导航的场景中，出行者只能依靠自身过往经验，根据经验记忆选择效用最大的出行模式和路径。这种场景下，出行者的每次决策都会得到环境带来的不同反馈，与强化学习的思想相契合。因此这种场景可以使用强化学

习的模型解决。

相较于传统的离散选择模型，强化学习存在以下三点优势：

1. 强化学习模型直接与环境交互，减少了传统离散选择模型的假设限制。传统离散选择模型需要对环境的条件预先假设并检验，在复杂多变的环境下传统模型的弊端将会体现。

2. 强化学习的模型会减少采集数据的成本。一项新的交通政策在实施前需要大量的仿真验证，传统模型需要采集大量的现实数据来验证模型的有效性。强化学习可以基于智能体已知的场景，通过更改仿真环境中的基础设施或策略，使得智能体学习处理未知场景下的决策行为。

3. 考虑智能体记忆能力，贴近实际决策过程。在强化学习的模型中，智能体做出动作后会根据以往经验以及自身的探索不断优化不同决策行为的价值及策略，这与实际中人在进行决策时的惯性一致。

1.4 论文的结构安排

第二章　深度强化学习相关知识

2.1　强化学习

强化学习是一种基于马尔可夫决策过程的算法，在规定的策略中，智能体根据现处的状态通过动作与环境进行交互，并产生新的状态，同时从环境得到奖励。重复循环此过程，直至智能体完成设定的目标[5]。强化学习算法就是利用在不断探索环境的过程中利用产生的奖励数据优化其行为策略, 以获得最大的回报。本节将首先介绍强化学习算法的相关术语。之后，根据智能体动作选取方式，将强化学习方法分为基于价值、基于策略，以及基于价值和策略三类分别综述。

2.1.1　相关术语

智能体：任何有独立思想并且可以与所处环境进行交互的实体。在交通场景下，智能体可以是行人，车辆，信号灯等。

状态：当前时刻智能体对周围环境的感知。所有时刻的感知集合构成了状态空间。

动作：智能体在当前状态下采取的行动。在当前状态下能采取的所有动作构成了动作空间。

策略：智能体根据当前时刻的状态应采取哪个动作的控制准则。在数学上的含义为，使用概率密度函数表示智能体在每个状态下采取各个动作的概率。

奖励：在智能体采取动作后，环境对智能体的反馈效果。奖励 R 可以为正反馈或负反馈。

回报：智能体从当前时刻至行动结束所能获得的累积奖励之和。

状态转移[6]：当智能体采取动作后，由当前状态转移到下一个状态的过程。状态转移过程大多数具有随机性，该随机性来源于环境。

2.1.2　基于价值的强化学习

基于价值的强化学习使智能体通过行动与奖励联系起来，通过试验和错误进行学习。智能体的主要目标是通过学习在不同情况下采取的最佳行动，随着时间的推移使其累积奖励最大化。在基于价值的强化学习中，智能体学习预测在特定状态下采取特定行动的价值。一个行动的价值通常被定义为智能体在特定状态下采取该行动并遵循特定政策所能获得的预期累积奖励。

在强化学习中，任意 t 时刻的状态 s_t 下执行策略 π 中的动作 a_t 都存在一个对应的奖励 R_t，由于强化学习所研究的问题具有马尔可夫性，因此系统的整体回报 U_t 与当前

时刻的奖励 R_t 和未来时刻的奖励 R_{t+n} 有关，所以存在等式：

$$U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ... + \gamma^n R_{t+n} \tag{2.1}$$

式中，γ 是折减因子。

在 t 时刻的回报 U_t 中，未来的奖励是与未来的状态和动作相关，而两者都具有随机性，所以需要通过对 U_t 求解期望值 $Q(s_t, a_t)$ 来消除随机性[7]。

$$Q(s_t, a_t) = E[U_t \mid S_t = s_t, A_t = a_t] \tag{2.2}$$

因此，$Q(s_t, a_t)$ 的值可以用来描述状态动作对 (s_t, a_t) 的价值。其中，$Q(s, a)$ 就是该强化学习的动作价值函数。通过寻找 t 时刻所有策略 π 的动作价值函数的最大值，可以得到最优的策略 π 的动作价值函数 $Q^*(s_t, a_t)$。

$$Q^*(s_t, a_t) = \max Q(s_t, a_t) \tag{2.3}$$

对最优策略 π 中的动作集 A 取最大值，即可获取每一次的最优动作 a^*。

$$a^* = \text{argmax}\, Q^*(s_t, a_t) \tag{2.4}$$

在基于价值的强化学习模型中，其主要目的就是逼近最优的策略 π 的动作价值函数 $Q^*(s_t, a_t)$。可以利用神经网络等方法近似动作价值函数进行求解。

由式2.4中动作价值函数 $Q^*(s_t, a_t)$ 可以得到价值最高的动作空间 A^*。在强化学习中，一般使用神经网络的方法近似函数 $Q^*(s_t, a_t)$，网络的输入为状态 s，网络的输出为不同动作的价值。则有：

$$Q(s, a; \mathbf{w}) \rightarrow Q(s, a) \tag{2.5}$$

式中，$\mathbf{w}$ 是价值网络 (Value Network) 的参数。可以通过不同状态下的奖励 R 利用时序差分算法更新价值网络，使得网络的参数 $\mathbf{w}$ 更加精确。

$$Q(s, a; \mathbf{w}) \approx R_t + \gamma \cdot Q(s, a; \mathbf{w}) \tag{2.6}$$

最常见的基于价值的强化学习算法是 Q-learning。Q-learning 是一种估计最佳动作价值函数的无模型方法，它代表了智能体在特定状态下采取特定动作并遵循最佳策略所能获得的预期累积奖励。一个状态-行动对的 Q 值使用贝尔曼方程进行更新，该方程指出，一个状态-行动对的最佳 Q 值等于即时奖励加上折现的最大预期未来奖励。Q-learning 是一个迭代过程，包括在智能体采取每个行动后更新 Q 值，并接受奖励形式的反馈。随着时间的推移，智能体学会了所有状态-行动对的最佳 Q 值，使其能够在每个状态下选择最佳行动，使其累积奖励最大化。

基于价值的强化学习方法已经在各种应用中取得了巨大的成功，包括游戏、机器人和自动驾驶汽车，使得智能体能够学习如何在复杂和不确定的环境中做出最佳决策。

2.1.3　基于策略的强化学习

基于策略的强化学习主要是为智能体在环境中采取行动寻找最佳策略，以使奖励最大化。策略是一种从状态到行动的映射，它告诉智能体在特定状态下应采取何种行动。基于策略的强化学习的目标是找到一个策略，使智能体的预期奖励在一段时间内最大化。在强化学习中，使用概率密度函数 $\pi(a \mid s)$ 来控制智能体在不同状态下的动作选取，即策略函数。策略函数的输入为当前 t 时刻的状态 s_t，输出为所有动作的概率值。依据策略函数得到的概率值对所有动作随机抽样后，确定在状态 s_t 下进行的动作 a_t。当使用神经网络的方法近似策略函数时，则有：

$$\begin{cases} \pi(a \mid s;\boldsymbol{\theta}) \rightarrow \pi(a \mid s) \\ \sum_{a \in A} \pi(a \mid s;\boldsymbol{\theta}) = 1 \end{cases} \tag{2.7}$$

式中，$\boldsymbol{\theta}$ 是策略网络的参数。

通过式2.2，对 $Q_\pi(s_t, a_t)$ 求取期望，通过积分消除概率密度函数 $\pi(\bullet|s)$ 中的动作 A 可以得到状态价值函数 V_π：

$$V_\pi(s_t) = E_A[Q_\pi(s_t, A)] \tag{2.8}$$

状态价值函数 $V_\pi(s_t)$ 只与当前策略 π 和状态 s_t 有关。因此，状态价值函数可以用来评价当前状态下不同策略的价值。如果是离散的动作空间，状态价值函数 $V_\pi(s_t)$ 可以写作：

$$V_\pi(s_t) = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a) \tag{2.9}$$

如果是连续的动作空间，则使用积分形式代替连加求和。由于连续动作空间的研究较复杂，并且大多数可以离散化，因此之后均为离散动作空间下的状态价值函数。通过式2.7中策略网络近似得到的策略函数 $\pi(a \mid s_t;\boldsymbol{\theta})$，可以近似状态价值函数：

$$V(s;\boldsymbol{\theta}) = \sum_a \pi(a \mid s;\boldsymbol{\theta}) \cdot Q_\pi(s_t, a) \tag{2.10}$$

基于策略的方法通常使用随机梯度上升法来更新策略。策略 π 由一组参数表示，使用策略梯度定理等技术计算出相对于这些参数的预期奖励的梯度。然后使用梯度上升法更新参数，以改进策略。策略学习是通过学习式2.9中的参数 $\boldsymbol{\theta}$，得到价值最高的策略。这个过程中需要通过不断地改进策略网络参数 $\boldsymbol{\theta}$ 的使 $V(s;\boldsymbol{\theta})$ 的值达到最大值。因此，可以将式2.9中的 $V(s;\boldsymbol{\theta})$ 对状态空间 S 求期望，将目标函数转化为 $J(\boldsymbol{\theta})$：

$$J(\boldsymbol{\theta}) = E_S[V(S;\boldsymbol{\theta})] \tag{2.11}$$

基于策略的强化学习的缺点之一是它的计算成本很高，因为策略通常由大量的参数表示。此外，策略有时会卡在局部最优处，这可能使它难以找到全局最优策略。

总的来说，基于策略的强化学习是一种在复杂和动态环境中寻找最优策略的强大方法，并且已经成功地应用于广泛的领域，包括机器人、游戏和自然语言处理。

2.1.4 价值与策略相结合的强化学习方法

在强化学习中，将策略网络与价值网络同时训练更新的方法称为策略价值结合学习方法。其目的为使智能体通过策略网络做出的动作价值越来越高的同时，使得价值网络对动作价值的评价越来越精准。在策略价值结合学习方法中，可以把策略网络当作行动者（actor），价值网络当作裁判（critic）。价值网络会对智能体通过策略网络做出的动作进行评价，帮助更新策略网络参数，

通过联立式2.5与式2.10，可以得到通过神经网络方法近似后的价值函数 $Q(s,a;\boldsymbol{w})$ 与策略函数 $\pi(a \mid s;\boldsymbol{\theta})$。因此，状态价值函数可以写作：

$$\begin{cases} V(s;\boldsymbol{\theta},\boldsymbol{w}) = \sum_a \pi(a \mid s;\boldsymbol{\theta}) \cdot q(s,a;\boldsymbol{w}) \\ \sum_{a \in A} \pi(a \mid s;\boldsymbol{\theta}) = 1 \end{cases} \tag{2.12}$$

此时，可以把策略网络当作行动者（actor），价值网络当作批评者（critic）。价值网络会对智能体通过策略网络做出的动作进行评价，帮助更新策略网络参数，使其目标函数 $J(\boldsymbol{\theta})$ 的值更大。行动者和批评者根据奖励和估计的状态-行动值进行更新。批评者通过最小化估计值和真实值（奖励和下一个状态的估计值之和）之间的平均平方误差来更新其对状态行动值的估计。行动者以估计的状态行动值为指导，通过最大化预期收益（未来奖励的总和）来更新其政策。

与其他强化学习算法相比，通过学习策略和价值函数，价值与策略相结合的强化学习方法可以比基于策略的方法更快地收敛，比基于价值的方法更稳定。它还可以处理高维的状态和行动空间，并且可以在实时环境中在线学习。价值与策略相结合的强化学习方法结合了基于政策和基于价值的方法的优点，可以同时学习最优政策和最优价值函数。然而，该方法需要仔细调整学习率和其他超参数以确保稳定的学习，而且它可能存在收敛问题和价值函数估计的偏差。

2.2 深度学习

2.2.1 神经网络

神经网络是一种机器学习模型，其灵感来自于人脑的结构和功能。它是由多个相互连接的节点或神经元组织成层，并形成一个系统。每个神经元接收来自其他神经元的输入，处理输入数据后产生一个输出信号。然后一个层的输出被用作下一层的输入，直至最终层输出结果。神经网络在训练期间从数据中学习经验并调整神经元之间连接的权重。权重决定了神经元之间的连接强度，它们使用优化算法进行更新，以达到预测输出和实际输出之间的误差最小化。神经网络已被应用于广泛的场景中，包括图像和语音识别、自然语言处理以及时间序列预测等。目前在深度学习领域内使用的主流神经网络结构有前馈神经网络、递归神经网络和卷积神经网络。

神经网络最常用的架构是前馈神经网络，其输入数据是沿同一个方向流经各层。前馈神经网络主体是由一个输入层、多个隐藏层和一个输出层组成。各个层的职责各不相

同：输入层接收输入数据，输出层产生神经网络的最终输出，隐藏层负责学习输入数据的特征。输入层的每个神经元代表输入数据的一个特征，而输出层的每个神经元代表神经网络预测的一个类别或一个值。隐蔽层中每个神经元的输出是通过对输入和权重的线性组合来计算的，并加入一个偏置项。然后，输出通过一个激活函数，将非线性引入网络。

隐藏层中每个神经元的输出可以按以下方式计算:

$$\begin{cases} z = \mathbf{w} * \mathbf{x} + b \\ a = f(x) \end{cases} \tag{2.13}$$

其中 z 是输入和偏置的加权和，$\mathbf{w}$ 是权重向量，$\mathbf{x}$ 是输入向量，b 是偏置项，$f(x)$ 是激活函数，a 是神经元的输出。

隐藏层中每个神经元的输出被用作下一层的输入, 在最后一层的输出是神经网络的最终输出。在训练过程中，神经网络通过调整神经元之间连接的权重和偏差，使预测输出和实际输出之间的差异最小。

反向传播是一种用于训练神经网络的算法，通过调整神经元之间连接的权重和偏置项来训练。它是一种基于梯度的优化算法，计算损失函数相对于权重和偏置的梯度，并按照负梯度的方向更新它们。其中，损失函数衡量的是预测输出和实际输出之间的误差。回归问题最常用的损失函数是平均平方误差，而分类问题则使用交叉熵损失。损失函数相对于权重和偏差的梯度可以用微积分的链式法则来计算。链式法则指出，一个复合函数的导数等于其组成部分的导数的乘积。在神经网络的背景下，链式法则被用来计算损失函数相对于每个神经元输出的导数，然后通过网络传播误差来调整权重和偏差。

损失函数对于神经元输出的梯度可以按以下方式计算。

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial z} * \frac{\partial z}{\partial a} \tag{2.14}$$

其中 L 是损失函数，a 是神经元的输出，z 是输入和偏置项的加权和。

式2.14右边的第一项是损失函数相对于加权和的导数，可以用激活函数的导数来计算。第二项是加权和相对于神经元输出的导数，也就是权重向量。然后，损失函数相对于权重和偏置的梯度可以通过在神经网络中各神经层向后传播误差来计算。首先计算输出层的误差，然后使用链式法则通过隐藏层向后传播，最后使用计算出的梯度和学习率更新权重和偏置项。权重通常在训练前被随机初始化。学习率决定了权重和偏置更新的步长，通常使用试验和错误或网格搜索来选择。高的学习率可能会导致对最优权重的过度拟合，而低的学习率则会导致缓慢的收敛。

训练神经网络的挑战之一是过拟合，即模型学习到的数据过于适合训练数据而在新数据上表现不佳。当模型相对于可用于训练的数据量来说过于复杂时，就会出现过拟合。正则化技术可以用来防止过度拟合。最常用的正则化技术是 L1 和 L2 正则化。L1 正则化给损失函数增加了一个惩罚项，与权重的绝对值成正比，而 L2 正则化则是增加

了一个惩罚项，与权重的平方成正比。惩罚项的作用是鼓励权重变小，这有助于防止过度拟合。

2.2.2 激活函数

激活函数是神经网络的一个关键组成部分，如果没有激活函数，神经网络本质上只是线性回归模型，这将严重限制其灵活性。激活函数是应用于神经网络中每个神经元的输出的函数，目的是在模型中引入非线性，这对于捕捉数据中的复杂模式来说是必要的。其计算过程主要是在神经元输入的加权和添加一个偏置项后，对结果进行非线性转换。之后，激活函数的输出值将被传递到网络的下一层作为输入数据。目前在深度学习中应用最广泛的激活函数有 Sigmoid 函数，ReLU 函数和 Softmax 函数。

Sigmoid 函数是一条平滑的 S 形曲线，接受任何输入并输出 0 到 1 之间的值。Sigmoid 函数的公式如下:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

其中 x 是该函数的输入。

Sigmoid 函数是可微的，这意味着它的导数可以在任何一点处计算出来，这使得它很适合用于反向传播，也就是用于训练神经网络的算法。此外，Sigmoid 函数在 0 和 1 之间是有界的，这意味着它可以被解释为一个概率。然而，Sigmoid 函数也有一些缺点。Sigmoid 函数的主要问题之一是它存在梯度消失的问题。当 Sigmoid 函数的输入非常大或非常小时，函数的输出分别变得非常接近于 0 或 1，函数的导数也会变得非常小，这可能导致梯度在反向传播期间消失。因为梯度在网络中向后传播时变得非常小，这样会使得网络难以学习更加深度的表征。

ReLU 函数是神经网络中另一个常用的激活函数。它是一个片状线性函数，接受任何输入，如果输入是正的，就输出，否则就是 0。ReLU 函数的公式如下:

$$f(x) = \max(0, x) \tag{2.16}$$

其中 x 是该函数的输入。

ReLU 函数的主要优点之一是它不存在梯度消失的问题。当 ReLU 函数的输入为正数时，该函数的导数为 1，这意味着在反向传播过程中，其计算出的梯度仍然很大。因为梯度在通过网络向后传播时不会消失，这使得网络更容易学习深度表征。ReLU 函数的另一个优点是它的计算效率高。由于该函数只是一个阈值操作，它可以用简单的逻辑运算来实现。然而，ReLU 函数的一个主要问题是，当 ReLU 函数的输入为负数时，该函数的输出为 0，这意味着神经元将会变得不活跃。这可能会导致整个神经元在训练过程中起不到任何作用，对网络的性能产生负面影响。

Softmax 函数是一种特殊的激活函数，常用于进行分类任务的神经网络的输出层。Softmax 函数的在接受到一个输入矢量后，会输出一个和为 1 的数值矢量，可解释为概

率。Softmax 函数由以下公式给出。

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.17}$$

其中 x_i 是输入矢量的第 i 个元素，和是在矢量的所有元素上取的。

Softmax 函数的主要优势是可以确保网络的输出被解释为概率。这使得它非常适合用于分类任务，其目标是将输入分配到几个可能的类别中的一个。此外，Softmax 函数是可微分的，这意味着它可以用于反向传播来训练网络。

除了上面讨论的激活函数外，还有许多其他类型的激活函数被用于神经网络，包括双曲正切函数、指数线性单元（ELU）函数和缩放指数线性单元（SELU）函数等。这些激活函数都有自己的特性和使用场景，激活函数的选择取决于被解决的问题的具体需求，不同的激活函数可能更适合于不同类型的数据或任务。通过了解不同激活函数的特性，可以更好地设计神经网络，使其能够捕捉到实际数据中存在的复杂模式。

2.2.3　损失函数及其优化算法

损失函数优化是训练神经网络的一个重要步骤。损失函数是用来衡量神经网络的预测输出和实际输出之间差异的数学函数。损失函数的目标是提供一个衡量神经网络表现如何的标准，损失函数优化是为了找到一组权重和偏置，使神经网络的预测输出与实际输出之间的差异最小。损失函数的选择取决于正在解决的具体问题。例如，对于回归问题，平均平方误差是一个常用的损失函数，而对于分类问题，通常使用交叉熵损失函数。

平均平方误差损失函数的公式如下：

$$L = \frac{1}{n} \cdot \sum_i y_i - \hat{y}_i{}^2 \tag{2.18}$$

其中 n 是数据集中的样本数，y_i 是第 i 个样本的实际输出，$\hat{y}_i$ 是第 i 个样本的预测输出，和是在数据集中的所有样本中取的。

交叉熵损失函数的公式如下：

$$L = -\frac{1}{n} \cdot \sum_i y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i) \tag{2.19}$$

其中 y_i 是第 i 个样本的实际输出（二元分类为 0 或 1，多类分类为单次编码向量），$\hat{y}_i$ 是第 i 个样本的预测输出（0 和 1 之间的概率），和是在数据集中的所有样本中取值。

优化算法的目标是找到使损失函数最小化的权重和偏置集合。神经网络最常用的优化算法是梯度下降法。梯度下降是一种迭代优化算法，它沿着损失函数的负梯度方向更新神经网络的权重和偏置。负梯度指向最陡峭的下降方向，这意味着沿着这个方向更新权重和偏置将导致损失函数值的减少。

梯度下降的更新规则如下：

$$w_i = w_i - \alpha \cdot \frac{\mathrm{d}L}{\mathrm{d}w_i} \tag{2.20}$$

其中 w_i 是第 i 个权重，α 是决定更新步长的超参数，dL/dw_i 是损失函数相对于第 i 个权重的偏导。

随机梯度下降（SGD）是梯度下降的一个变种，它基于随机选择的小型训练数据子集来更新权重和偏差。这样做的好处是在计算上比梯度下降法更有效率，因为它只需要计算一小部分数据的梯度。随机梯度下降的更新规则与梯度下降相似，但使用在数据子集上计算的梯度而不是整个数据集, 更新规则：

$$w_i = w_i - \alpha \cdot \frac{\mathrm{d}L_i}{\mathrm{d}w_i} \tag{2.21}$$

其中 $\mathrm{d}L_i/\mathrm{d}w_i$ 是损失函数相对于当前数据子集的第 i 个权重的偏导。

总之，损失函数优化是深度学习的一个关键方面，因为它直接影响到神经网络的性能。通过使用各种优化算法，如梯度下降、SGD 和 Adam，我们可以训练我们的神经网络来最小化损失函数，并在各种任务中获得高精确度。

2.3 深度强化学习

2.3.1 深度 Q 网络

深度 Q 网络（DQN）是基于 Q-learning 算法的深度强化学习算法，其目的是使用深度神经网络来近似给定状态-动作对的最佳动作价值函数。在 Q-learning 中，智能体通过选择最大化动作价值 Q 的行动来学习最大化其预期的未来回报，Q 值是在给定状态下采取行动并在之后遵循给定策略的预期未来回报。

在式2.2和式2.3中分别给出了动作价值和最佳动作价值的定义，DQN 算法使用一个深度神经网络来近似动作价值函数。该神经网络将当前状态作为输入，为每个可能的行动输出一个动作价值，智能体所选动作的动作价值被用来更新神经网络的权重。在 DQN 算法中，下一个状态的动作价值是用目标网络来估计的，目标网络是一个具有固定权重的主网络的复制模型。目标 Q 值被用来更新主网络的权重，主网络被用来估计当前状态的动作价值。用于更新神经网络权重的损失函数 $L(\theta)$ 目的是估计动作价值和目标动作价值之间的平均平方误差：

$$L(\theta) = E\left[\left(r + \gamma \cdot \max_{a'} Q(s', a', \theta') - Q(s, a, \theta)\right)^2\right] \tag{2.22}$$

其中，θ 是主网络的权重，θ' 是目标网络的权重，r 是在状态 s 下采取行动 a 后获得的即时奖励。

DQN 算法使用经验回放来提高学习效率和稳定性。经验回放存储了一个固定大小的经验缓冲区，神经网络通过从缓冲区中随机抽取经验进行训练。经验重放减少了连续经验之间的相关性，使学习过程更加有效。经验重放也有助于防止网络对最近的经验过度拟合。

DQN 算法使用 epsilon-greedy 探索来平衡对新行动的探索和对当前策略的利用。Epsilon-greedy 探索以 1-ϵ 的概率选择具有最高 Q 值的行动，以 ϵ 的概率选择一个随机行动。

$$a_t = \begin{cases} \underset{a \in A}{\text{argmax}} Q\left(s_t, a\right) & \text{概率为} 1-\varepsilon, \\ \text{rand}(a) & \text{概率为} \varepsilon. \end{cases} \tag{2.23}$$

总之，DQN 算法是一种强大的、广泛使用的方法，用于训练强化学习问题中的动作价值函数。它通过使用神经网络来近似动作价值函数，解决了传统 Q-learning 算法的一些局限性，这使得它可以在类似的状态和行动中进行泛化。它还使用了一个经验重放缓冲器和一个目标网络来提高稳定性并防止过度拟合。DQN 算法已被成功应用于各种具有挑战性的强化学习问题，包括游戏、机器人和控制。通过了解 DQN 算法背后的原理，研究人员和从业人员可以将其有效地应用于新的问题领域，并继续推进深度强化学习的技术水平。

2.3.2　近端策略优化

近端策略优化 (PPO) 是一种近年来备受关注的深度强化学习策略优化算法。与其他深度强化学习算法相比，PPO 算法具有易于实现、高效稳定等优点。本文将介绍 PPO 算法的基本原理、优化目标函数、算法流程以及实现细节。近似策略优化（PPO）是一种强化学习算法，属于策略梯度方法家族。与其他政策梯度方法不同，PPO 因其简单和有效而受到欢迎。它被广泛用于各种应用，包括游戏、机器人和自动驾驶汽车。PPO 是一种政策上的算法，这意味着它从使用当前政策收集的经验中学习。

PPO 算法的基本原理是通过优化策略函数来寻找最优策略。在强化学习中，策略函数通常是一个映射函数，它将当前状态作为输入，输出对应的行动。PPO 算法通过反复迭代，不断更新策略函数，使其逐渐趋于最优。PPO 算法的核心思想是限制每次策略更新的大小，避免策略函数发生大幅度变化，导致训练不稳定。具体而言，PPO 算法采用一种被称为“近端策略优化”的方法，通过在优化目标函数中增加一个约束项来限制每次策略更新的大小。这个约束项通常被称为“剪切项”，它会限制新旧策略之间的差异，并确保每次策略更新的大小不超过一个预设的阈值。PPO 算法的优化目标函数通常被定义为最大化一个期望回报函数。具体而言，假设我们要最大化的期望回报函数为 $J(\theta)$，其中 θ 表示策略函数的参数。在 PPO 算法中，优化目标函数可以被表示为以下形式：

$$\max_{\theta} \mathbb{E}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A^{\pi_{\theta_{old}}}(s,a)\right] - \epsilon \cdot KL\left(\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s)\right)$$

其中，$\pi_\theta(a|s)$ 表示当前策略下在状态 s 下选择行动 a 的概率；$\pi_{\theta_{old}}(a|s)$ 表示上一轮迭代中的策略下在状态 s 下选择行动 a 的概率；$A^{\pi_{\theta_{old}}}(s,a)$ 表示状态 s 下采取行动 a 相对于当前价值函数的优势值；ϵ 表示剪切项的系数，$KL(\cdot,\cdot)$ 表示 KL 散度。

目标函数的第一项表示策略更新的目标是最大化期望回报函数，第二项表示对策略更新进行剪切，确保新策略不会偏离原来的分布太远。通常来说，ϵ 的取值较小，可以取 0.1 或 0.2 等较小的数值。当 ϵ 取较小值时，第二项的影响较小，策略更新更倾向于最大化期望回报函数。

PPO 算法的优点在于它能够克服许多其他算法的缺点，如 TRPO 算法的收敛速度较慢和 DDPG 算法的稳定性较差。通过采用近端策略优化的方法，PPO 算法可以避免策略函数更新过于迅速，导致训练不稳定的问题。同时，PPO 算法具有高效的计算性能，可以在 GPU 上快速训练复杂的深度神经网络。

PPO 算法的流程通常可以分为四个步骤：采样、计算优势函数、更新策略函数和更新价值函数。下面我们将分别介绍这些步骤的具体内容。

在 PPO 算法中，采样是一个非常重要的步骤。采样过程需要从当前策略中采样一定数量的样本数据，并用这些数据更新策略函数和价值函数。为了实现样本的高效采集，通常使用并行采样的方法，在多个并行的环境中同时采集数据，以提高采样效率。

优势函数表示采取某个行动相对于当前价值函数的优势程度，是 PPO 算法中计算策略更新目标函数的重要组成部分。在 PPO 算法中，优势函数可以表示为以下形式：

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

其中，$Q^{\pi}(s,a)$ 表示状态 s 下采取行动 a 的价值函数，$V^{\pi}(s)$ 表示状态 s 的价值函数。优势函数表示了当前策略相对于最优策略的优劣程度，可以用来指导策略函数的更新。在 PPO 算法中，更新策略函数的方法通常是通过梯度下降法来实现。具体而言，PPO 算法采用近端策略优化的方法，将目标函数表示为以下形式其中，ϵ 是 PPO 算法中的一个超参数，通常取一个较小的值，如 0.1 或 0.2 等。$\pi_{\theta}(a_t|s_t)$ 表示策略函数在状态 s_t 下选择行动 a_t 的概率，$\pi_{\theta_{old}}(a_t|s_t)$ 表示旧的策略函数在状态 s_t 下选择行动 a_t 的概率，r_t 表示在状态 s_t 下采取行动 a_t 后获得的回报，A_t 表示在状态 s_t 下采取行动 a_t 的优势函数，$V_{\theta}(s_t)$ 表示在状态 s_t 下的价值函数，c_1 和 c_2 是两个超参数，通常取 0.5。在目标函数中，第一项是 PPO 算法的约束项，用于限制新的策略函数与旧的策略函数之间的 KL 散度，保证策略函数的更新不会太大。第二项是 PPO 算法的优化项，用于最大化期望回报和优势函数的加权和。PPO 算法通过在 KL 散度约束下最大化优化项来更新策略函数，从而实现稳定的训练过程。

在 PPO 算法中，更新价值函数的方法通常是通过均方误差（MSE）损失函数来实现。具体而言，PPO 算法采用类似于 DQN 算法的目标网络更新方法，将目标函数表示为以下形式：

$$L_{vf}(\theta) = \frac{1}{2}\mathbb{E}s_t \sim \mathcal{D}, r_t \sim \mathcal{N}(0,\sigma^2)[(V\theta(s_t) - r_t - V_{targ}(s_t))^2]$$

其中，$V_{targ}(s_t)$ 表示目标价值函数，在 PPO 算法中通常采用 TD 目标法来计算，即 $V_{targ}(s_t) = r_t + \gamma V_{\theta}(s_{t+1})$。$\mathcal{D}$ 表示经验回放缓冲区，用于存储历史采样数据。PPO 算法

通过最小化 MSE 损失函数来更新价值函数，从而提高策略函数的准确性。

PPO 算法是一种高效、稳定的深度强化学习算法，已经在多个实际应用场景中得到了广泛的应用。与其他深度强化学习算法相比，PPO 算法具有以下优点：

稳定性好：PPO 算法采用近端策略优化的方法，避免了策略函数更新过于迅速导致训练不稳定的问题。同时，PPO 算法通过 KL 散度约束和剪切参数等技术，有效地控制了策略函数的更新幅度，进一步提高了算法的稳定性。

收敛速度快：PPO 算法通过近端策略优化的方法，避免了传统强化学习算法中存在的梯度爆炸和梯度消失等问题，从而能够更快地收敛到最优解。适用范围广：PPO 算法可以应用于各种强化学习场景，包括连续动作空间和离散动作空间，同时还能够应用于多智能体协同决策、强化学习任务转移等场景。

然而，PPO 算法也存在一些不足之处。例如，PPO 算法对于大规模离散动作空间的问题处理较为困难，同时其算法复杂度较高，需要消耗大量的计算资源。此外，PPO 算法在处理一些特殊场景下，如存在不确定性的环境、存在噪声的环境等，可能会出现训练不稳定的问题。

综上所述，PPO 算法是一种高效、稳定的深度强化学习算法，已经在众多实际应用场景中得到了广泛的应用。在使用 PPO 算法时，需要根据具体场景进行超参数的调整，从而获得最优的训练效果。

2.3.3　深度确定性策略梯度

Deep Deterministic Policy Gradient（DDPG）算法是一种结合了深度神经网络和确定性策略梯度（Deterministic Policy Gradient）的强化学习算法。DDPG 算法在连续动作空间的问题上取得了很好的效果，同时在许多实际应用场景中得到了广泛的应用。

DDPG 算法主要用于解决连续动作空间的问题，这类问题中，智能体需要在一个连续的动作空间中选择动作，因此传统的强化学习算法无法直接应用于该类问题。DDPG 算法通过结合深度神经网络和确定性策略梯度的方法，解决了这一问题。与传统的 Q-Learning 算法相比，DDPG 算法在处理连续动作空间问题时，可以直接输出动作值，而无需在离散动作空间中搜索最优动作。

DDPG 算法主要由四个部分组成：Actor 网络、Critic 网络、经验回放缓存和目标网络。其中 Actor 网络和 Critic 网络都采用深度神经网络来进行参数化。

Actor 网络的作用是输出在当前状态下最优的动作值。Critic 网络的作用是评估 Actor 网络输出的动作值的优劣，同时根据这个评估值来更新 Actor 网络。经验回放缓存的作用是记录智能体在环境中的经验，并从中随机采样用于网络的训练。目标网络的作用是解决训练不稳定的问题，其参数是由 Critic 网络参数每隔一段时间拷贝而来的。

DDPG 算法的主要思路是通过 Actor-Critic 模型来学习动作值函数，同时通过确定性策略梯度的方法来更新策略函数。具体来说，DDPG 算法在训练时首先需要通过经验回放缓存来收集一定数量的状态转移样本，然后从中随机采样一批样本，用于网络的训

练。每次更新 Actor 网络时，通过 Critic 网络评估 Actor 网络的动作值，然后根据这个评估值计算出相应的策略梯度，最后通过反向传播算法更新 Actor 网络的参数。每次更新 Critic 网络时，需要计算出 TD 误差，然后通过反向传播算法更新 Critic 网络的参数。在更新完 Actor 和 Critic 网络后，需要更新目标网络的参数，从而保证网络的稳定性。

DDPG 算法的优点主要体现在以下几个方面：

可以直接处理连续动作空间问题：DDPG 算法通过 Actor-Critic 模型，可以直接输出动作值，因此可以直接处理连续动作空间问题。训练稳定：DDPG 算法通过目标网络的引入和经验回放缓存的使用，可以使网络的训练更加稳定。

可以处理高维状态空间问题：DDPG 算法使用深度神经网络来处理高维状态空间问题，可以有效地提取状态特征信息，从而提高智能体的决策效果。可以处理具有连续动作空间和延迟奖励的问题：DDPG 算法结合了确定性策略梯度和 Q-Learning 的思想，可以同时处理具有连续动作空间和延迟奖励的问题。DDPG 算法的缺点主要包括以下几个方面：

训练时间长：DDPG 算法中需要使用经验回放缓存和目标网络等技术来提高训练的稳定性，但这也导致了训练时间的增加。参数调节复杂：DDPG 算法中有许多超参数需要调节，例如网络结构、学习率、优化器等，这些超参数的不同取值会影响算法的表现。对噪声敏感：DDPG 算法在处理连续动作空间问题时，通常需要引入噪声，但对噪声的选择和调节会对算法的表现产生较大影响。总体而言，DDPG 算法是一种在连续动作空间问题上表现优秀的强化学习算法。它结合了深度神经网络和确定性策略梯度的思想，可以直接处理连续动作空间问题，同时可以处理高维状态空间和延迟奖励的问题。虽然 DDPG 算法在训练时间长、参数调节复杂和对噪声敏感等方面存在一些缺点，但它在许多实际应用场景中得到了广泛的应用。

第三章　仿真实验场景的设计与构建

3.1　城市交通仿真平台的可行性分析

3.1.1　Vissim 介绍

VISSIM（VISual SIMulation）是由德国 PTV（Planung Transport Verkehr AG）公司开发的一款功能强大的交通模拟软件。它被广泛应用于交通规划、设计、管理等领域。VISSIM 可以模拟各种复杂的交通场景，包括城市道路、高速公路、交叉口、公共交通，以及交通流、交通信号控制、公共交通路线规划、车辆路径规划等。VISSIM 的基本功能包括：

1. 建模。VISSIM 的建模功能非常强大。用户可以使用 VISSIM 的图形用户界面轻松创建交通场景，如道路、交叉口和公共交通路线。用户还可以调整交通流率、车辆类型、行人流量和其他参数，以建立一个交通网络。在 VISSIM 的智能交通生成器的帮助下，可以快速、准确地创建交通场景。交通生成器使用真实的交通数据，为一天中的不同时段、工作日和周末创建真实的交通流模式。交通生成器还允许用户定制交通情景，并调整交通量、速度和其他参数。

2. 仿真模拟。VISSIM 提供了一个高保真的交通仿真引擎，可以高度准确地模拟各种交通场景。用户可以运行各种交通管理策略的模拟，如交通信号控制、公共交通路线规划和车辆路径规划。仿真引擎还可以生成实时交通数据，如车辆速度、行驶时间和延迟时间。有了这些数据，用户可以评估不同交通管理策略的性能，比较不同的方案，并做出明智的决定。

3. 分析功能。VISSIM 提供了一个强大的分析功能，使用户能够分析和可视化模拟结果。用户可以生成大量的图表和表格来分析交通流量、拥堵情况、车辆行驶时间、车辆速度和其他指标。分析功能还允许用户比较不同交通管理策略的性能，评估不同参数对交通流的影响。

VISSIM 是一个全面而强大的交通模拟软件，为用户提供准确而真实的模拟结果。该软件支持各种类型的交通场景，从小型交叉口到大型城市网络。建模功能允许用户创建一个虚拟的交通网络，模拟引擎产生的实时交通数据可用于评估不同的交通管理策略和交通计划。该软件的高级功能，如定制、多模式模拟、行人模拟、排放和油耗以及驾驶辅助系统，为用户提供了一套全面的工具来评估交通网络的性能和影响。总之，VISSIM 是交通专业人员、研究人员和政策制定者设计、分析和优化交通系统的重要工具，以实现更安全、更高效和可持续的未来。

3.1.2 SUMO 介绍

SUMO（Simulation of Urban Mobility）是一款开源的交通仿真软件，被广泛应用于城市交通规划、交通管理、交通研究等领域。SUMO 是由德国科研机构 DLR 开发，其设计目标是实现高效、可扩展和高度自定义的交通仿真，同时提供良好的可视化和数据输出。SUMO 具有开放性、高可靠性、高可扩展性和良好的可视化效果等特点，成为了交通仿真领域的重要工具之一。

SUMO 的基本功能包括：

1. 建模。SUMO 提供了强大的建模功能，允许用户创建一个虚拟交通网络。该软件支持各种道路网络，包括高速公路、公路、交叉口和环岛等。用户可以定义网络中的道路布局、交通流和车辆类型。此外，该软件还提供了从各种文件格式（如 OpenStreetMap）导入道路网络的工具。

2. 仿真技术。SUMO 使用微观交通模拟引擎，提供准确和真实的模拟结果。该软件可以模拟各种交通场景，如车辆路线、公共交通和行人运动。仿真引擎能够生成实时交通数据，如旅行时间、车辆速度和延迟时间。这些数据可以用来评估不同的交通管理策略和交通计划。

3. 可视化。SUMO 提供了一个用户友好的图形用户界面，允许用户对模拟结果进行可视化。该界面提供了一个交通网络的实时视图，并可以显示个别车辆、行人和公共交通的运动。此外，该软件还提供了生成各种图表和表格的工具，以总结模拟结果。

SUMO 的高级功能有：

1. 定制功能。SUMO 是高度可定制的，允许用户定义影响模拟的不同参数，如车辆类型、司机行为和交通信号。该软件还提供了一个脚本接口，允许用户通过编写自定义脚本来扩展软件的功能。用户还可以创建自定义的车辆模型，并将其导入到模拟中。

2. 多模式仿真。SUMO 支持多模式模拟，允许用户模拟多种交通方式，如汽车、公交车、火车和自行车。这个功能可以评估不同交通方式的性能，并对不同的交通计划进行比较。

3. 优化功能。SUMO 提供优化功能，使用户能够改善交通网络的性能。该软件可以优化交通信号灯时间、车辆路线和公共交通时间表。优化功能允许用户找到最佳的交通管理策略和交通计划。

4. 并行化。SUMO 提供并行化功能，允许用户在多个处理器上运行模拟。这个功能加快了模拟时间，并能模拟更大的交通网络。

5. 集成。SUMO 可以与其他软件工具集成，如交通流模型、地理信息系统（GIS）和数据分析工具。集成功能使数据的交换和定制解决方案的开发成为可能。

总之，SUMO 是一个功能强大、用途广泛的模拟软件，为用户提供了一套全面的工具来模拟和分析各种交通情况。该软件支持多模式交通的模拟，包括汽车、公交车、自行车和行人，它还提供了一些高级功能，如交通需求生成、交通信号控制和车辆路由。SUMO 的灵活架构和开源代码库使其成为研究人员、交通专业人士和需要高度可定制

模拟工具的决策者的热门选择。SUMO 有能力生成真实的交通数据，并提供对交通系统性能的洞察力，在设计、优化和评估交通系统方面发挥了关键作用，以实现更安全、更高效和可持续的未来。

3.1.3　Matsim 介绍

MATSim 是一个开源的、基于代理的、多模式的交通模拟软件，为城市或地区的个人行为和整个交通系统的建模和模拟提供了一个平台。该软件由瑞士苏黎世联邦理工学院（ETH Zurich）的一个研究团队历经数年开发，目前已被世界各地的研究人员、交通规划人员和政策制定者广泛使用。

MATSim 提供了一套全面的功能来模拟和仿真不同的交通场景。它将个人旅行者建模为代理人，根据他们的偏好和现有的交通选择做出决定。代理人可以选择不同的交通方式，如汽车、公共交通、步行或骑自行车，他们还可以选择最短、最快或最舒适的路线来到达目的地。该软件的先进模拟引擎可以生成大规模的模拟，可以在任何标准计算机系统上运行。

MATSim 软件基于模块化架构，允许用户根据自己的具体需求定制软件。这些模块可以被扩展或被其他模块取代，以模拟新的运输系统或包括新的功能。这一特点使该软件能够适应不同交通场景的具体要求，并使其成为研究人员和交通专业人士的热门选择。

MATSim 的主要特点之一是它能够模拟多模式的交通系统。它可以模拟广泛的交通选择，包括汽车、公共汽车、火车、自行车和步行，并且可以模拟不同交通方式之间的相互作用。这使用户能够评估不同交通系统的性能，并优化城市或区域内不同交通方式的使用。

MATSim 还包括一个全面的可视化工具，允许用户实时查看模拟的结果。该软件的可视化模块使用户能够看到模拟的代理人在交通网络中移动，做出决定，并到达他们的目的地。可视化模块还可以显示模拟的交通流量、旅行时间和其他重要指标，可用于评估交通系统的性能。

MATSim 的另一个主要特点是其开源代码库。该软件是在 GNU 通用公共许可证下发布的，它允许用户修改软件并将其分发给其他人。这一特点使研究人员和交通专业人士能够合作和分享他们的工作，使 MATSim 成为交通规划和管理领域的研究和开发项目的热门选择。

此外，MATSim 软件已经由一个研究人员和交通专业人士团队进行了广泛的测试和验证。该软件已被用于对全球不同城市和地区的交通系统进行建模和模拟，包括苏黎世、柏林、悉尼、新加坡等。这有助于建立该软件的可信度和可靠性，并证明其对交通系统进行精确建模和模拟的能力。

总之，MATSim 是一款功能强大、用途广泛的交通仿真软件，为用户提供了一套全面的工具来模拟和仿真不同的交通场景。该软件的模块化架构、多模式模拟能力、先进

的可视化功能和开源代码库使其成为研究人员和交通专业人士的热门选择。MATSim 能够准确地模拟交通系统并评估不同交通方案的性能，在设计、优化和评估交通系统以实现更安全、更高效和可持续的未来方面发挥了关键作用。

3.1.4 仿真平台的选择

在这三个交通模拟软件中，我会选择 SUMO，原因有很多。

首先，SUMO 是一个开源软件，这意味着它可以免费下载和使用。这对于预算紧张的人来说是一个很大的优势，因为使用 SUMO 没有任何许可费用。此外，SUMO 的开源性质意味着它是高度可定制的，这使得用户可以根据他们的具体需求来修改软件。

其次，SUMO 是一个高度通用的软件，可以模拟多模式的交通场景。它可以模拟汽车、公共汽车、自行车、行人，甚至火车。这意味着用户可以对各种交通场景进行建模和模拟，并评估不同交通方案的性能。

第三，SUMO 是一个高度精确的模拟软件。该软件以先进的交通流和车辆动力学模型为基础，提供高度真实和准确的结果。这使得 SUMO 成为交通专业人士、研究人员和政策制定者的重要工具，他们需要评估交通系统的性能，并优化它们，以实现更安全、更高效和可持续的未来。

第四，SUMO 有一个友好的界面，这使得初学者和高级用户都很容易使用。该软件的界面允许用户创建和修改交通网络，生成交通需求，并评估不同交通系统的性能。SUMO 还提供了一套全面的可视化工具，使用户可以实时查看模拟结果。

第五，SUMO 有一个庞大而活跃的用户社区，它提供了丰富的资源，包括教程、文档和用户论坛。这意味着，如果用户在使用该软件时遇到任何问题，可以很容易地找到支持和帮助。此外，用户社区还提供了一个合作和分享最佳实践和新发展的平台。

最后，SUMO 是交通行业中备受推崇和广泛使用的交通仿真软件。它已被用于模拟和仿真世界上许多城市和地区的交通系统，包括欧洲、亚洲和北美。这意味着 SUMO 已经被交通专业人员广泛测试和验证，其结果也被政策制定者和决策者所信任。

总之，SUMO 是一个强大的、多功能的交通仿真软件，为用户提供了一套全面的工具来模拟和仿真不同的交通场景。它的开源性、多模式模拟能力、准确性、用户友好的界面、活跃的用户社区和受人尊敬的声誉使它成为交通专业人士、研究人员和决策者的热门选择。凭借其准确模拟交通系统和评估不同交通方案性能的能力，SUMO 在设计、优化和评估交通系统以实现更安全、更高效和可持续发展的未来方面发挥了关键作用。

3.2　基于 SUMO 的城市交通仿真平台

3.2.1　平台设计目标

3.2.2　功能模块简介

3.3　实验场景的选择与搭建

3.3.1　路网的编辑与生成

3.3.2　出行模式的设计

3.3.3　流量的生成

第四章　基于深度强化学习的出行模式与时间选择方法

强化学习是一种通过迭代地改进策略来最大化累计奖励或回报的机器学习方法。在应用强化学习到具有马尔可夫属性的序列决策过程中，需要先构建一个马尔可夫决策过程，该过程定义了环境的演变，考虑到强化学习代理所采取的行动。强化学习代理通过行动探索和开发不断地与环境互动，根据当前状态 s_t 进行行动。每次行动会使环境演变成一个新状态 s_{t+1}，并获得相应的奖励 r_t，反馈给代理以改善其决策逻辑。这个过程一直迭代，直到代理成功学习到一个能够最大化累计奖励的策略 π，也就是一个决策者。因此，强化学习的关键在于根据奖励不断迭代改进策略。

在本研究中，将每个出行者视为具有学习能力的智能实体，通过马尔可夫决策过程来建模每个出行者跨越多个连续日的交通出行行为。每个出行者能够选择的行动包括不同组合的出行方式和出发时间。最终由个人采取的行动 a_t 决定了环境演变到的下一个状态 s_{t+1}。该状态应反映个人关于行程本身以及相关环境的最新知识。选择此行动所获得的奖励 r_t（与旅行成本相关）有助于改善个人的决策逻辑，这样个人就能逐渐学习到最优的行动策略，并最大化累计奖励。

4.1　马尔可夫决策过程框架

正如先前所讨论的那样，马尔可夫决策决策过程是建模和优化出行模式和时间选择的先决条件。从数学角度来说，它是一个五元组 (S, A, P, R, γ)，其中 S 表示状态空间，A 表示动作空间，P 表示状态转移概率，R 表示奖励函数，最后 γ 是折扣因子。在这里，智能体是出行模式和时间选择中个人的决策者。虽然将每个个体视为智能体在概念上是有效的，但是由于代理数量等于个体数量，所产生的计算复杂性是大规模应用的主要障碍。从推荐系统的角度来看，研究适用于所有个体的常见决策逻辑是值得探究的，这反映了该方法的普适性。接下来，我们将进一步阐述如何构建和解决问题特定的马尔可夫决策决策过程。

4.1.1　动作空间

动作空间是强化学习中的一个关键概念。在建模动作空间时需考虑出行者的个性化特征。例如，不同出行者对于出行方式和出发时间的偏好不同，因此他们的动作空间也会不同。对此，可以引入个性化因素对动作空间进行建模。例如，可以考虑出行者的年龄、性别、职业、家庭状况等因素，进一步细化动作空间的描述，提高模型的预测能力和适应性。

此外，动作空间的大小和粒度也会影响到模型的性能和可解释性。如果动作空间过

大，模型的训练和预测会变得非常困难，同时也会增加模型的计算复杂度和存储空间需求。而如果动作空间过小，模型的表达能力就会受到限制，无法对真实情况进行有效建模。因此，需要在合理范围内对动作空间进行定义和限制，以平衡模型的性能和可解释性。在实际应用中，建模动作空间的过程也需要考虑到数据的可用性和质量。例如，在收集出行调查数据时，需要尽可能全面和准确地收集出行者的出行方式和出发时间等信息，以便更好地建模动作空间和进行模型预测。同时，在对数据进行预处理和清洗时，也需要对动作空间进行适当的定义和筛选，以避免噪声和异常数据对模型的影响。

在 JTMDTC 中，动作空间包括出行方式和出发时间两个方面。在选择出发时间时，出行者需要考虑到交通拥堵、出行时间和其他因素对行程的影响。例如，在高峰期出发可能需要更长的旅行时间，而在非高峰期出发可能可以更快地到达目的地。因此，在建模动作空间时，需要综合考虑各种因素，以便在代理决策时提供准确的信息。出行方式可以是私家车、公共交通或自行车。在公共交通中，可以选择乘坐公交车或地铁，但是不考虑三种交通方式之间的换乘。这是因为换乘涉及到很多变量，比如停车地点、换乘时间等，需要更多的研究来处理。

对于出发时间，每个出行者都有一个初始或期望出发时间 t_0。但是，由于各种原因，可能需要调整出发时间，比如交通拥堵、天气等。在 JTMDTC 中，出发时间可以在一个时间窗口 $[t_{\text{min}}, t_{\text{max}}]$ 内进行调整。这个时间窗口是由最早和最晚的出发时间 t_{min} 和 t_{max} 确定的。出发时间的调整是以离散间隔为单位进行的，而不是连续方式进行。这是因为在实际交通中，时间通常是以分钟为单位进行的，而不是连续的时间。

对于上述所提到的交通方式和出发时间，需要进行适当的编码以便于代理在模型中进行操作。在本文中，采用离散化编码的方式，将交通方式和出发时间分别离散化为一组离散的选项。例如，对于交通方式，可以将私家车、公交车、地铁和自行车分别编码为 m_1、m_2、m_3 和 m_4。对于出发时间，可以将 t_0 和时间窗口 $[t_{\text{min}}, t_{\text{max}}]$ 离散化为一组时间步长，例如每 5 分钟一步。这样，代理可以从一组离散的选项中进行选择，并决定最佳的出行方式和出发时间。动作空间的描述采用了向量的形式。向量 $\boldsymbol{a}$ 包括交通方式 $\tilde{m}$ 和出发时间 $\tilde{t}$。交通方式可以是可用交通方式 $m_1, m_2, \ldots, m_N$ 中的任意一种，而出发时间必须在时间窗口 $[t_{\text{min}}, t_{\text{max}}]$ 内。因此，动作空间可以表示为：

$$\boldsymbol{a} = \begin{bmatrix} \tilde{m} \\ \tilde{t} \end{bmatrix} = \begin{bmatrix} \text{出行模式} \\ \text{出发时间} \end{bmatrix} \in \begin{bmatrix} \{m_1, m_2, \ldots, m_N\} \\ [t_{\text{min}}, t_{\text{max}}] \end{bmatrix}, \tag{4.1}$$

动作空间是模型中重要的一部分，它定义了代理能够采取的所有行动，直接影响着模型的效果和性能。在建模时，需要综合考虑多种因素，将交通方式和出发时间等重要信息进行适当的编码，以便于代理进行决策。

4.1.2 状态空间

状态空间定义了智能体选择行动的上下文环境。对于 JTMDTC，状态空间被设计为不仅包含有关行程的最新知识，还包括智能体早期经验。这种状态空间设计在很大程度

上类似于理性人类的决策机制，即从经验中学习。由于重点不在于经验选择建模而在于选择指导或推荐，因此我们假设智能体能够充分感知环境，因此对行程具有完全信息。然而，我们稍后将讨论这种假设可能会有所放松。

行程信息首先包括每种交通方式的旅行距离 L 和记忆旅行时间 $\bar{T}$。前者是模式 m 的旅行距离，而后者是模式 m 的平均经验旅行时间。其原因有两个——利用经验和在交通随机性存在的情况下保持稳健性。作为行程信息的另外两个变量是初始出发时间 t_0 和出发时间差或偏移量 Δt 相对于 t_0。将所有内容组合起来，得到以下特定于行程信息的状态向量：

$$\boldsymbol{s}_{\text{trip}}^{m} = \left[\begin{array}{cccc} L_m & \bar{T}m & t0 & \Delta t \end{array}\right] = \left[\begin{array}{cccc} \text{旅行距离} & \text{经验旅行时间} & \text{初始出发时间} & \text{出发时间差} \end{array}\right]. \tag{4.2}$$

环境信息基本上包括有助于不同交通方式旅行成本的因素。对于公共交通，考虑到两个因素，即可达性和票价。在这里，我们将可达性 p 定义为完成行程的第一和最后一段所需的总步行距离：

$$p = d_{\text{origin}} + d_{\text{destination}}, \tag{4.3}$$

其中 d_{origin} 和 $d_{\text{destination}}$ 分别是从最近的公交车站或地铁站到起点和终点的步行距离。公共交通票价是必须支付的使用该服务的货币成本。对于私家车，我们考虑燃油价格作为影响因素，并将其放在状态中。因此，特定于环境信息的状态向量如下所示：

$$\boldsymbol{s}_{\text{reduced}} = \left[\begin{array}{c} \bar{T} \\ t_0 \\ \Delta t \end{array}\right] = \left[\begin{array}{c} \text{memory travel time} \\ \text{initial departure time} \\ \text{departure time difference} \end{array}\right]. \tag{4.4}$$

最后，我们要注意到，上述状态向量（或状态空间）中包含的变量可能因特定问题或应用而异。状态空间设计的目标是确保代理可以获得对其行动选择有意义的环境信息。对于某些应用程序，可能需要包括更多的环境因素，例如天气和道路条件。而对于其他应用程序，可能只需要少数几个变量即可获得有效的行动建议。因此，状态空间设计取决于特定的问题和应用场景。

总之，状态空间是强化学习中非常重要的一个概念，它定义了代理在决策时需要考虑的上下文环境。在设计状态空间时，我们需要仔细考虑应用场景和问题的特征，以确保状态空间中包含的信息能够有效地指导代理的决策。同时，我们也需要注意状态空间的维度和大小，以便使代理能够有效地处理状态，并且可以在有限的时间内完成状态的学习和更新。

4.1.3 奖励函数

4.2 基于深度 Q 网络算法的模式与出发时间选择算法

4.2.1 神经网络结构设计

The first step in designing a DQN model is to choose an appropriate neural network architecture.

The architecture should be capable of taking the state as input and producing Q-values for each action in the action space as output.

Convolutional neural networks (CNNs) are commonly used for problems with image-based inputs, while fully connected neural networks can be used for problems with vector-based inputs.

The number and size of layers in the network can also vary depending on the complexity of the problem.

4.2.2 超参数的选择与标定

Hyperparameters play a crucial role in the performance of a DQN model.

The learning rate, batch size, discount factor, and other hyperparameters should be carefully selected to optimize the performance of the model.

The learning rate controls the rate at which the model updates its weights during training.

The batch size determines how many state-action pairs are processed in each training iteration.

The discount factor determines the balance between immediate and future rewards in the Q-value calculation.

Other hyperparameters, such as the size of the replay buffer or the frequency of target network updates, can also have a significant impact on model performance.

4.2.3 模型的优化

Various optimization techniques can be used to improve the stability and convergence of a DQN model.

Experience replay is a technique where past experiences are stored in a buffer and sampled randomly during training.

Target networks can be used to estimate the Q-values of the next state to improve stability during training.

Prioritized experience replay can be used to sample more important experiences more frequently.

Reward shaping can be used to modify the reward function to incentivize certain behaviors.

The selection and use of these optimization techniques can depend on the specific problem and the network architecture used.

4.3 模型的训练与评估

4.3.1 数据收集与预处理

Before training a DQN model, it is important to preprocess the input data to make it suitable for the neural network.

This can include steps such as normalization, scaling, and cropping of images.

The RL agent also needs to collect data by interacting with the environment.

During this process, the agent chooses actions based on the current state, receives a reward, and transitions to the next state.

The state, action, reward, and next state are stored in a replay buffer, which is then used for training.

4.3.2 模型训练

To train a DQN model, the neural network is updated using stochastic gradient descent with mini-batches of state-action pairs from the replay buffer.

During each training iteration, the model predicts Q-values for the current state, and the Q-value for the chosen action is compared to the target Q-value, which is calculated using the Bellman equation.

The difference between the predicted and target Q-values is used to calculate the loss, which is then backpropagated through the network to update the weights.

To improve stability and convergence, various techniques such as experience replay, target networks, and prioritized experience replay can be used.

4.3.3 模型的评估

Once the DQN model is trained, it is evaluated to assess its performance.

The agent interacts with the environment using the trained model, and the cumulative reward obtained over a fixed number of episodes is used as a metric of performance.

The agent may also be tested on a holdout set of data to assess its generalization capabilities.

If the performance of the model is not satisfactory, the hyperparameters or model architecture can be adjusted, and the training and evaluation process can be repeated.

第五章　针对多智能体的模式与出发时间选择方法

5.1　基于聚类的深度强化学习方法

5.1.1　DBSCAN 聚类方法

5.1.2　聚类参数的选择

5.1.3　深度强化学习模型的改进

5.2　模型的验证

5.2.1　与传统方法的对比

5.2.2　模型参数的灵敏性分析

第六章　总结与展望

6.1　总结

本模板基于宋睿同学发布在SEU-master-thesis 并在上述工作的基础上进行了微调，解决了一些自己编写代码过程中 BUG。

6.2　展望

致　谢

感谢许元和樊智猛等前人的工作，没有他们的工作也就不会有这个模板的诞生。也感谢使用该模板的每一个人，因为你们的开放与进取心使得 LaTeX 在东南大学的氛围越来越好。

参考文献

[1] 李梦凡. 交通信息诱导下个体出行选择行为研究[D]. 合肥工业大学, 2018.

[2] McFadden D, et al. Conditional logit analysis of qualitative choice behavior[C]. Frontiers in Econometrics. 1973. 105-142.

[3] de Jong G, Daly A, Pieters M, et al. A model for time of day and mode choice using error components logit[J]. Transportation Research Part E: Logistics and Transportation Review, 2003, 39(3):245-268.

[4] Fukuda D, Yai T. Semiparametric specification of the utility function in a travel mode choice model[J]. Transportation, 2010, 37(2):221-238.

[5] 陈学松，杨宜民. 强化学习研究综述[J]. 计算机应用研究, 2010(2834-2838+2844).

[6] 高阳，陈世福，陆鑫. 强化学习研究综述[J/OL]. 自动化学报, 2004(86-100). DOI: 10.16383/j.aas.2004.01.011.

[7] 李茹杨，彭慧民，李仁刚，赵坤. 强化学习算法与应用综述[J/OL]. 计算机系统应用, 2020(13-25). DOI: 10.15888/j.cnki.csa.007701.

参考文献

[1] 李梦凡. 交通信息诱导下个体出行选择行为研究[D]. 合肥工业大学, 2018.

[2] McFadden D, et al. Conditional logit analysis of qualitative choice behavior[C]. Frontiers in Econometrics. 1973. 105-142.

[3] de Jong G, Daly A, Pieters M, et al. A model for time of day and mode choice using error components logit[J]. Transportation Research Part E: Logistics and Transportation Review, 2003, 39(3):245-268.

[4] Fukuda D, Yai T. Semiparametric specification of the utility function in a travel mode choice model[J]. Transportation, 2010, 37(2):221-238.

[5] 陈学松，杨宜民. 强化学习研究综述[J]. 计算机应用研究, 2010(2834-2838+2844).

[6] 高阳，陈世福，陆鑫. 强化学习研究综述[J/OL]. 自动化学报, 2004(86-100). DOI: 10.16383/j.aas.2004.01.011.

[7] 李茹杨，彭慧民，李仁刚，赵坤. 强化学习算法与应用综述[J/OL]. 计算机系统应用, 2020(13-25). DOI: 10.15888/j.cnki.csa.007701.

作者简介

知心哥哥（1996.3.3 -），男，台湾南昌人，现居台湾重庆，为网易 CC 丢人主播，主要研究方向有《黑旗》、《黑楼》和《黑暗剑》。

作者攻读硕士学位期间发表的论文

[1]. **ZHi X**, WEI T, CHEN R, et al. Sea of Thieves: Fucking Animal[C]. 2017 810th International Conference on Disgraced (ICD). Chongqing, 2017. 1-6. (EI Indexed)

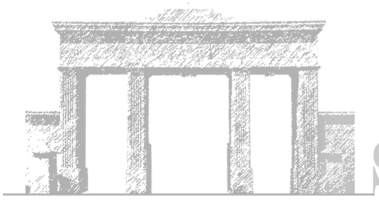
[2]. **ZHi X**, WEI T, JI H, et al. Animal Crossing: Playing Together[J]. Nintendo Daily Journal, 2020, 13(3): 114-514. (SCI Indexed)

[3]. 韦天, **知心哥哥**. 你怪猎来我大圣：3D 游戏之耻 [C]. 第 19 届口吐芬芳游戏评测国际会议 (SFGR). 台湾南昌, 2019. 1-14. (EI Indexed)

作者攻读硕士学位期间参与的研究课题

[1]. **2018.5-2019.2**：底特律便乘人暨强人工智能的实现

[2]. **2020.1-2020.3**：大老爹拿球的概率模型研究



SOUTHEAST UNIVERSITY