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。

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 函数图像如图2-1所示。

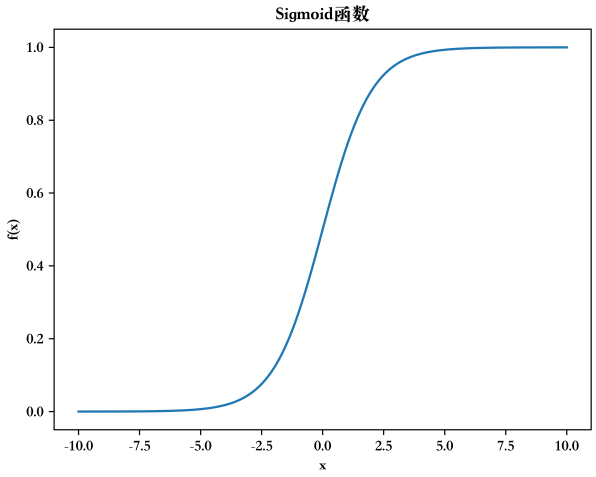


图 2-1 Sigmoid 函数图像

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:

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

ReLU 函数图像如图2-2所示。

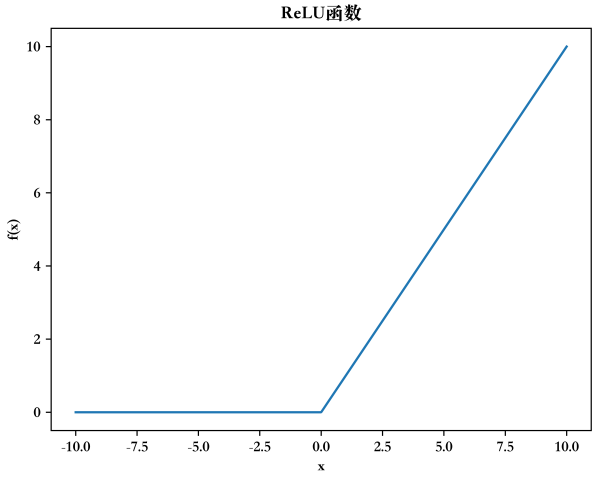


图 2-2 ReLU 函数图像

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 函数图像如图2-3所示。

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

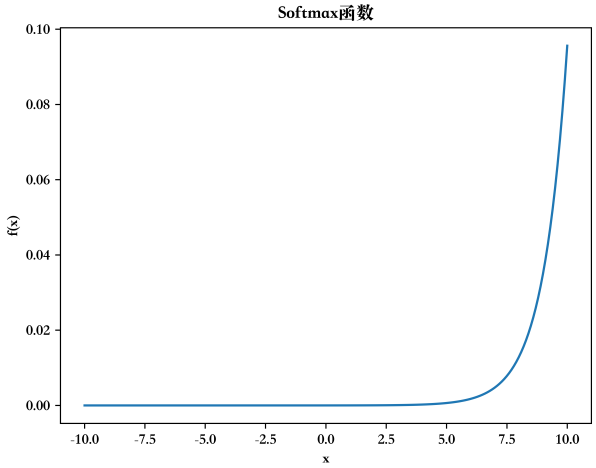


图 2-3 Softmax 函数图像

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，其目标是提供一个衡量神经网络表现如何的标准。而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，通常使用平均平方误差，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。

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。DQN 中使用的平均平方误差损失函数，它用于衡量网络输出的 Q 值和目标 Q 值之间的差距。可根据式2.18得到用于更新神经网络权重的损失函数 $L(\theta)$:

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

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。

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

总之，DQN 算法是一种被广泛使用的强化学习方法，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决策问题中，包括游戏、机器人的控制等。

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, 属于策略梯度方法，这意味着它将从当前策略收集的经验中学习。PPO 算法的基本原理是通过优化策略函数来寻找最优策略。通过式2.7可以得知，在强化学习中，策略函数 $\pi(a \mid s)$ 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。

该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，以确保优化的稳定性。在 PPO 算法中，优化目标函数是最大化经过剪切后的期望优势函数，而不是最大化期望回报函数。这里的优势函数表示当前策略相对于旧策略的性能提升程度。在 PPO 算法中，我们使用剪切方法来限制当前策略和旧策略之间的差异，其优化目标函数可以写作：

$$L_{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) A_t \right) \right] \tag{2.24}$$

其中，θ 表示当前策略的参数，θ_{old} 表示旧策略的参数，$\pi_\theta(a_t|s_t)$ 表示在状态 s_t 下，当前策略选择动作 a_t 的概率，$\pi_{\theta_{\text{old}}}(a_t|s_t)$ 表示在状态 s_t 下，旧策略选择动作 a_t 的概率，A_t 表示在时刻 t 的优势函数，用于表示在给定状态 s_t 和行动 a_t 下，相比于平均水平的预期奖励，当前策略的表现优劣程度。具体地，A_t 的定义如下：

$$A_t = Q(s_t, a_t) - V(s_t) \tag{2.25}$$

式2.24中，目标函数 $L_{\text{CLIP}}(\theta)$ 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。

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，类似于式2.22。PPO 算法已经在多个实际应用场景中得到了广泛的应用，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。

2.3.3　深度确定性策略梯度

深度确定性策略梯度（DDPG）算法是一种结合了深度神经网络和确定性策略梯度的强化学习算法，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。其主要由四个部分组成：策略网络、价值网络、经验回放缓存和目标网络。其中策略网络和价值网络都采用深度神经网络来进行参数化，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。策略网络的作用是输出在当前状态下最优的动作值。价值网络的作用是评估策略网络输出该动作值优劣的评估值，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策略网络的参数。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，其参数是由价值网络参数每隔一段时间拷贝而来的。

DDPG 算法的主要优点可以总结为以下几个方面。首先，DDPG 算法采用 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。其次，DDPG 算法引入目标网络和经验回放缓存技术，可以提高网络的训练稳定性。此外，DDPG 算法使用深度神经网络处理高维状态空间问题，可以有效提取状态特征信息，提高智能体的决策效果。最后，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。

然而，DDPG 算法也存在一些缺点：1. 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的同时会导致了训练时间的增加。2.DDPG 算法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，参数调节较为复杂。3.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，对噪声较为敏感。

2.3.4 深度强化学习算法的对比与选择

在解决出行模式和时间选择问题时，本章已经介绍了三种深度强化学习算法：深度 Q 网络、近端策略优化和深度确定性策略梯度。现在需要对这三种算法进行对比，并最终确定选择哪种算法。

表 2.1 三种深度强化学习算法的对比

模型	优点	
深度 Q 网络	训练速度快、稳定性高	收敛较慢，对
近端策略优化	收敛速度快，能够保持高样本效率	训练时间较长
深度确定性策略梯度	能够处理高维状态空间和延迟奖励；能够学习到高质量的策略	对参数的选择

DQN 算法是一种基于 Q-Learning 算法和深度神经网络的强化学习算法。其优点是训练速度快、稳定性高，而且适用于离散动作空间和连续动作空间。缺点是难以处理连续状态空间问题。在解决出行模式和时间选择问题中，我们的状态空间较大，因此 DQN 算法的表现可能会受到限制。

PPO 算法是一种基于策略梯度的强化学习算法。其优点是能够处理连续状态空间和连续动作空间问题，具有很好的表现。缺点是训练速度较慢，而且需要大量的超参数调节。

DDPG 算法是一种基于 Actor-Critic 模型和深度神经网络的强化学习算法。其优点是能够直接处理连续动作空间问题和高维状态空间问题，而且能够处理具有延迟奖励的问题。缺点是训练时间较长，且对超参数的调节比较敏感。

综上所述，针对出行模式和时间选择问题，我们最终选择了 DQN 算法。因为我们的问题是一个离散的动作空间，而且 DQN 算法的训练速度快、稳定性高，适合处理这种问题。同时，我们也需要考虑到算法的可解释性和易于实现性，DQN 算法在这方面也具有优势。虽然 PPO 算法在处理连续状态空间和连续动作空间问题上表现较好，但在我们的问题中，状态空间较大，训练速度也较慢，不太适合。DDPG 算法可以处理连续动作空间和高维状态空间问题，但训练时间较长，且对超参数的调节比较敏感，因此也不太适合我们的问题。因此，我们最终选择了 DQN 算法来解决出行模式和时间选择问题。

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

本章旨在介绍使用 SUMO 仿真平台进行基于深度强化学习的出行模式和时间选择研究的设计目标。我们将首先概述本研究的研究背景和目的，然后介绍 SUMO 仿真平

台的优点和特点，并说明我们将如何使用 SUMO 进行仿真实验，最后总结本章内容。

本研究旨在通过深度强化学习方法，探究出行者的出行模式和时间选择行为，并通过对不同出行者的学习，实现出行者之间的智能交互和协同，从而优化交通流量和提高出行效率。为此，我们选择 SUMO 仿真平台作为我们的仿真工具，以模拟不同的交通环境和交通管理策略，以及出行者之间的交互和协作。

SUMO 仿真平台是一个开源的、高度可定制的交通流量仿真器，可以模拟不同的交通场景和交通管理策略。它提供了一个完整的仿真工具链，包括道路网络编辑器 Netedit、仿真器 SUMO、交互式仿真器 SUMO-GUI 和命令行接口 TraCI 等。SUMO 支持多种车辆类型和行驶策略，如私家车、公共交通、自行车和行人等。同时，SUMO 还支持多种路线选择算法和交通灯控制算法，用户可以选择最适合他们应用场景的算法。

在我们的研究中，我们将使用 SUMO 仿真平台来创建不同的交通环境和交通管理策略，并在其中加入深度强化学习算法来模拟出行者的行为。我们将使用 SUMO-Tools 和 SUMO-Plugins 等工具来分析和比较不同的交通管理策略和出行者行为模型，以评估其效果和优化方案。我们还将使用 TraCI 接口来控制和管理仿真场景，以模拟不同的交通环境和交通流量。

综上所述，本章介绍了使用 SUMO 仿真平台进行基于深度强化学习的出行模式和时间选择研究的设计目标。我们选择 SUMO 作为我们的仿真工具，以模拟不同的交通环境和交通管理策略，并使用深度强化学习算法来模拟出行者的行为。我们将使用 SUMO-Tools 和 SUMO-Plugins 等工具来分析和比较不同的交通管理策略和出行者行为模型，以评估其效果和优化方案。最终，我们希望通过本研究探究出行者之间的智能交互和协同，以优化交通流量和提高出行效率。通过使用 SUMO 仿真平台，我们可以创建复杂的交通场景，并添加自定义的功能和行为，以满足特定应用程序的需要。同时，SUMO 还提供了丰富的数据输出和可视化功能，可以方便地对仿真结果和统计数据进行分析和可视化，为我们的研究提供了重要的支持和帮助。

在本章中，我们还将介绍我们的仿真平台设计的一些具体要素和技术细节，如如何设置仿真场景、如何定义出行者的行为模型、如何评估不同的交通管理策略等。我们将详细阐述如何使用 SUMO-Tools 和 SUMO-Plugins 等工具来分析和比较不同的交通管理策略和出行者行为模型，以及如何使用 TraCI 接口来控制和管理仿真场景。

综上所述，本章介绍了使用 SUMO 仿真平台进行基于深度强化学习的出行模式和时间选择研究的设计目标。我们选择 SUMO 作为我们的仿真工具，以模拟不同的交通环境和交通管理策略，并使用深度强化学习算法来模拟出行者的行为。通过本章的介绍，我们希望读者能够了解我们的仿真平台设计目标和技术细节，为后续章节的研究工作打下基础。

3.2.2 功能模块简介

Netedit 是 SUMO 中一个重要的功能模块，它允许用户可视化创建、编辑和管理道路网络。Netedit 提供了一个交互式的图形界面，用户可以通过鼠标和键盘输入，轻松地添加和编辑道路、路口、车道、交通灯等元素。Netedit 支持多种地图格式，如 OpenStreetMap、Google Maps 和 Bing Maps 等。用户可以直接导入现有的道路网络或数据，也可以通过 Netedit 创建新的道路网络。

在 Netedit 中，用户可以通过简单的拖放操作和线条绘制工具添加道路和路口。用户可以定义每条道路的长度、车道数、最大速度和路权等信息。路口可以包括单向或双向转向，用户可以设置交通灯的定时和调度策略。用户还可以定义车辆行驶的规则和路权，以模拟不同的交通环境和交通管理策略。同时，Netedit 还提供了一些有用的工具，如道路长度和宽度测量工具，帮助用户更加准确地绘制道路网络。

在创建和编辑道路网络之后，用户可以导出 Netedit 文件以供 SUMO-GUI 或其他仿真器使用。SUMO-GUI 可以可视化地显示道路网络和交通流量，以便用户进行交通流量仿真和结果分析。Netedit 还支持 Python 脚本，用户可以通过编写脚本来自动化地创建和编辑道路网络。这为大规模道路网络的创建和管理提供了便利。

综上所述，Netedit 是 SUMO 中一个强大的道路网络编辑器，它提供了一个易于使用和直观的图形界面，允许用户轻松地创建、编辑和管理道路网络。通过 Netedit，用户可以定义道路、路口、车道、交通灯等元素，以模拟不同的交通环境和交通管理策略。同时，Netedit 还支持多种地图格式和 Python 脚本，为用户提供了更多的定制和扩展能力。

TraCI 是 SUMO 中一个重要的功能模块，它允许用户创建和控制车辆和路口的运动和行为。TraCI 可以通过 Python 脚本进行调用和控制，这使得用户可以模拟复杂的交通环境和交通管理策略。TraCI 支持多种车辆类型和行驶策略，如私家车、公共交通、自行车和行人等。

TraCI 提供了一系列 API 接口，用于访问和修改仿真器中的车辆和路口状态。用户可以查询车辆的位置、速度、加速度、目标路段等信息，并控制车辆的加速、刹车和转向。用户也可以控制路口的红绿灯和车辆进出等操作。TraCI 还提供了一些有用的工具和功能，如路网查询、车辆生成和路由规划等，帮助用户更好地控制和管理交通流量。

TraCI 可以与其他 SUMO 模块和工具集成，如 SUMO-GUI、SUMO-Tools 和 SUMO-Plugins 等。这使得用户可以创建复杂的交通管理策略和仿真场景，以模拟不同的交通环境和交通流量。同时，TraCI 还支持可视化输出和数据记录，用户可以轻松地分析仿真结果和统计数据。

综上所述，TraCI 是 SUMO 中一个功能强大的交通流量生成器，它允许用户创建和控制车辆和路口的运动和行为。通过 TraCI，用户可以查询和修改车辆和路口状态，控制交通灯和车辆进出等操作，以模拟不同的交通管理策略和交通环境。TraCI 还支持多种车辆类型和行驶策略，并可以与其他 SUMO 模块和工具集成，为用户提供更多的定

制和扩展能力。

SUMO-Tools 是 SUMO 中一个重要的功能模块，用于模拟和评估交通管理策略，如交通灯控制和路由优化。SUMO-Tools 包括一个流量分析器、一个行驶速度分析器和一个决策支持工具等。用户可以使用 SUMO-Tools 分析和比较不同的交通管理策略，以评估其效果和优化方案。

SUMO-Plugins 是 SUMO 的一个可扩展模块，允许用户添加自定义的功能和行为。它们可以是 Python 脚本、C++ 插件或 Java 插件等。用户可以使用 SUMO-Plugins 来创建和添加新的车辆类型、路线选择器、行驶策略和交通管理策略等。SUMO-Plugins 还支持多种路线选择算法和交通灯控制算法，用户可以选择最适合他们应用场景的算法。

SUMO-Tools 和 SUMO-Plugins 可以与其他 SUMO 模块和工具集成，如 SUMO-GUI 和 TraCI 等。用户可以使用 SUMO-GUI 可视化地显示仿真结果和统计数据，使用 TraCI 控制和管理仿真场景。同时，用户可以编写 Python 脚本和 C++ 插件等扩展 SUMO-Plugins，以满足特定应用程序的需要。这些功能模块和工具为用户提供了更多的定制和扩展能力，使得 SUMO 成为一个强大的交通模拟器。

综上所述，SUMO-Tools 和 SUMO-Plugins 是 SUMO 中两个重要的功能模块，用于模拟和评估交通管理策略和扩展 SUMO 的功能和行为。它们可以与其他 SUMO 模块和工具集成，为用户提供更多的定制和扩展能力。通过使用 SUMO-Tools 和 SUMO-Plugins，用户可以模拟和评估不同的交通管理策略，创建复杂的交通场景，并添加自定义的功能和行为，以满足特定应用程序的需要。

3.3　实验场景的选择与搭建

3.3.1　路网的编辑与生成

在本研究中，我们使用 SUMO（Simulation of Urban MObility）作为交通仿真引擎。SUMO 是一个开源的交通仿真软件，支持建模、仿真和分析各种交通场景，包括道路交通、公共交通、自行车交通和行人交通等。

为了进行基于深度强化学习的出行模式和时间选择研究，我们需要建立一个仿真环境来模拟交通流。在选择仿真区域方面，我们选择了中国苏州市的一个城市区域作为我们的仿真区域，该区域面积大约为 20 平方公里。这个区域是一个典型的城市区域，具有复杂的交通结构和各种出行方式。因此，这个区域是一个理想的研究对象，可以帮助我们更好地了解基于深度强化学习的出行模式和时间选择。

为了建立仿真环境，我们使用了 OpenStreetMap（OSM）获取网络的几何和配置信息（2,423 个节点和 4,970 条边），并将这些信息输入到 SUMO 中进行仿真。OSM 是一个免费的、开源的地图服务，可以提供全球范围内的地图数据，包括道路、建筑、自然环境等。我们可以利用 OSM 提供的地图数据，轻松地建立仿真环境。

由于我们的仿真区域是一个多模式交通网络，公共交通（包括公交和地铁）也必须

被配置。为了实现这一点，我们首先从地图服务应用程序中提取公共交通运营信息，然后进行地图匹配。提取的信息包括线路 ID、停靠站或车站 ID 及其地理位置。这些信息可以通过地图匹配技术与实际地图中的位置进行匹配，从而实现公共交通在仿真环境中的配置。

最终，我们成功建立了一个具有复杂交通结构和多种出行方式的仿真环境，可以用于研究基于深度强化学习的出行模式和时间选择。通过 SUMO 的仿真功能，我们可以对不同出行模式和时间选择的影响进行量化分析，帮助我们更好地了解这些问题。

3.3.2 出行模式的设计

出行模式的设计是交通仿真中的一个重要方面。在本研究中，我们旨在研究深度强化学习在出行模式和出发时间选择方面的应用。因此，我们需要在 SUMO 交通仿真软件中建立一个具有多种出行方式的仿真环境。

在本研究中，我们考虑了私家车、公共交通（包括公交车和地铁）和自行车三种出行方式。私家车是最常见的出行方式之一，而公共交通和自行车则是城市交通中的重要组成部分。为了实现这种多模式交通网络的仿真，我们需要对这些出行方式进行适当的建模和参数化。

首先，我们需要确定私家车、公共交通和自行车在 SUMO 仿真中的特征和参数。对于私家车，我们需要考虑其速度、加速度和刹车距离等因素。对于公共交通，我们需要考虑车辆的数量、停靠站点、运营时间和乘客人数等因素。对于自行车，我们需要考虑其速度、加速度和行驶距离等因素。

其次，我们需要将这些特征和参数纳入到 SUMO 仿真中，以便代理可以在仿真环境中对它们进行操作。对于私家车，我们可以利用 SUMO 的 Car-Following 模型来建模其运动行为。对于公共交通，我们可以利用 SUMO 中的公交车和地铁模型来建模其运营行为。对于自行车，我们可以利用 SUMO 的 Bicycle 模型来建模其运动行为。通过将这些模型组合在一起，我们可以实现多种出行方式的仿真。

最后，我们需要在 SUMO 仿真环境中建立一个多模式交通网络。这个网络需要包括私家车、公共交通和自行车在内的所有出行方式，以便代理可以在仿真环境中对它们进行操作。我们需要将道路、交叉口和停靠站点等元素纳入仿真环境中，并通过 SUMO 的网络编辑工具对其进行编辑和配置。这样，我们就可以建立一个具有多种出行方式的仿真环境，用于研究基于深度强化学习的出行模式和出发时间选择。

在本研究中，我们成功地建立了一个具有复杂交通结构和多种出行方式的仿真环境，可以用于研究出行模式和出发时间选择。通过 SUMO 的仿真功能，我们可以对不同出行模式和出发时间选择的影响进行量化分析，帮助我们更好地了解这些问题。同时，我们还可以通过仿真来测试不同的出行政策和规划方案，以便为决策者提供科学的依据和参考。

为了建立融合公共交通、私家车和自行车的多模式交通网络，我们需要对网络进行

适当的编辑和生成。在本研究中，我们使用 OpenStreetMap（OSM）来获取网络的几何和配置信息，并将这些信息输入到 SUMO 中进行仿真。在 OSM 中，道路、路径、建筑和其他空间要素都被准确地绘制在地图上。SUMO 可以将这些信息转换为交通仿真模型，并在模拟中使用它们。同时，我们还需要配置公共交通（包括公交车和地铁）和自行车的路线和停靠点。这些信息可以从地图服务应用程序中提取，并与 OSM 中的信息进行匹配。

在多模式交通网络中，不同的出行模式需要不同的道路和路径支持。例如，自行车需要有专门的自行车道，而公共交通需要有专门的公交车道和地铁隧道。在网络生成过程中，我们需要将这些要素考虑在内，并将其配置到对应的道路和路径上。这样，在模拟中，不同的出行模式可以根据自己的需求使用不同的道路和路径，从而更好地模拟现实世界中的交通流。

在多模式交通网络中，出行者的出行方式选择涉及到多个变量和因素。例如，私家车出行者可能会考虑到停车地点和停车费用，而公共交通出行者可能会考虑到车站或站点的位置和交通运行时间等。在模型设计中，我们需要考虑到这些因素，以便为出行者提供准确的出行方式选择。这些因素可以编码为模型的状态变量，并在代理决策时提供给代理。

总之，通过在 SUMO 中建立融合公共交通、私家车和自行车的多模式交通网络，我们可以更好地研究出行模式和出发时间选择问题。这种仿真方法可以提供高度可控的实验环境，并帮助我们更好地理解交通流的动态变化和出行者的行为决策。

3.3.3　流量的生成

为了进行交通仿真实验，我们需要生成一定的交通流量。本研究中，我们选择典型的早高峰时段（上午 7 点至 9 点）作为研究对象，考虑了五个连续工作日内的出行模式和时间选择。每个工作日被视为训练过程中的一个 episode。虽然我们使用了合成的旅行需求，但这并不影响所提出方法的有效性。仿真时段为 2 小时，被分为 4 个 30 分钟的时间间隔。对于每个工作日，我们有一个典型的早高峰出行需求模式，对于每个时间间隔，出行需求被视为一个遵循高斯分布的随机变量（见表 1）。因此，在仿真过程中，每个 episode 都与略有不同的出行需求相关联。这种随机性符合现实中观察到的某种模式下的交通流量日常波动（即周期性拥堵）。然而，在非周期性拥堵或意外事件的情况下，所提出的方法可能不再适用，因为这些事件对出行选择的影响没有被体验和学习。在这方面进行更进一步的研究是值得的。

我们使用 OpenStreetMap（OSM）获取网络的几何和配置信息（2,423 个节点和 4,970 条边），并将这些信息输入到 SUMO 中进行仿真。同时，由于我们的仿真区域是一个多模式交通网络，因此我们需要配置公共交通（包括公交和地铁）。为了实现这一点，我们首先从地图服务应用程序中提取公共交通运营信息，然后进行地图匹配。提取的信息包括线路 ID、停靠站或车站 ID 及其地理位置。这些信息可以通过地图匹配技术与实际

地图中的位置进行匹配，从而实现公共交通在仿真环境中的配置。

表 1 展示了我们对五个连续工作日内早高峰时段的交通需求配置。表格中列出了每个时间段的车辆数，以及使用的随机分布。我们将每个工作日的交通需求视为随机变量，通过高斯分布模拟早高峰期间的交通需求波动。这种方式可以使得每个 episode 的交通需求略有不同，从而更贴近实际的交通流量波动情况。

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：

$$\boldsymbol{s}_{\text{trip}}^{m} = \begin{bmatrix} L_m \\ \bar{T}_m \\ t_0 \\ \Delta t \end{bmatrix} = \begin{bmatrix} \text{travel distance} \\ \text{memory travel time} \\ \text{initial departure time} \\ \text{departure time difference} \end{bmatrix}. \tag{4.2}$$

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：

$$\boldsymbol{s}_{\text{reduced}} = \begin{bmatrix} \bar{T} \\ t_0 \\ \Delta t \end{bmatrix} = \begin{bmatrix} \text{memory travel time} \\ \text{initial departure time} \\ \text{departure time difference} \end{bmatrix}. \tag{4.4}$$

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

在强化学习中，智能体的目标是通过最大化长期奖励来学习最优的决策规则。通过奖励函数，智能体可以计算每个动作对于实现这个目标的预期收益。在这个例子中，我们的奖励函数是旅行效用，旨在最小化总旅行费用。智能体将根据预期的长期奖励来选择动作，以便在未来的交互中最大化收益。

当智能体选择并执行一个动作时，会导致环境从当前状态转移到一个新状态。同时，智能体还会收到一个反馈或奖励，旨在改善智能体的决策逻辑。奖励可以是正面的，意味着动作是明智的，也可以是负面的，意味着相反。在这里，我们定义奖励为旅行效用，这主要由各种货币成本组成。具体来说，步骤 i 获得的奖励计算如下：

$$r_i = \frac{E_1 - C_m^i}{E_2}, \tag{4.5}$$

其中，C_m^i 是交通方式 m 的总旅行费用，E_1 和 E_2 是映射和缩放成本到奖励的两个常数。总旅行费用又可以分解为三个部分，即总旅行时间 T_m^i、行程延误 $\delta(t^i)$ 和其他与旅行相关的成本 F_m^i。

$$C_m^i = \alpha T_m^i + \delta\left(t^i\right) + F_m^i, \tag{4.6}$$

其中，α 是时间的价值。这个公式描述了在一个旅行过程中，奖励是如何被计算的，以及成本是如何被划分的。智能体可以通过调整其动作来优化它所接收到的奖励，并在行程中实现更好的效用。

只考虑总旅行时间的问题是忽略了实际到达时间。也就是说，尽管总的旅行时间很短，但到达时间可能与理想的时间相差甚远。因此，引入了计划延迟，这个概念可以追溯到[8]。假设每个人都有一个期望的到达时间，早到和晚到都会产生一个所谓的日程延误成本。当实际到达时间偏离期望时间时，计划延迟成本会以线性方式增长。从数学上讲，它表示为：。

$$\delta(t^i) = \begin{cases} \beta\left(t + T_m - t_{\mathrm{d}}\right) & \text{if} \quad t + T_m - t_{\mathrm{d}} < 0, \\ 0 & \text{if} \quad t + T_m - t_{\mathrm{d}} = 0, \\ \gamma\left(t_{\mathrm{d}} - t - T_m\right) & \text{if} \quad t + T_m - t_{\mathrm{d}} > 0, \end{cases} \tag{4.7}$$

其中，β 和 γ 分别是早到和晚到的行程延误单价，t_{d} 是期望到达时间。通过考虑行程延误成本，可以更准确地评估各种出行方式的效用。

除此之外，其他的与出行相关的费用主要是指汽车的燃料费用和公共交通的票价。对于私家车来说，燃料费用是与行驶距离成正比的，而对于公共交通来说，则是根据具

体的交通工具的票价。因此，其他出行相关费用可以表示为公式：

$$F_m^i = \begin{cases} L_{\text{car}} \cdot o & \text{if} \quad m = \text{car}, \\ f & \text{if} \quad m = \text{public transportation}, \\ 0 & \text{if} \quad m = \text{cycling}, \end{cases} \tag{4.8}$$

其中，f 可以表示为：

$$f = \mathbb{I}(\text{ bus }) \cdot f_{\text{bus}} + \mathbb{I}(\text{ subway }) \cdot f_{\text{subway}}, \tag{4.9}$$

这里的 $\mathbb{I}(\cdot)$ 是一个指示函数，当选择的交通工具是公共汽车或地铁时，它返回 1，否则返回 0。公交车费用 f_{bus} 是固定的，而地铁费用 f_{subway} 则随着行驶距离的增加而增加。这些费用可以用于计算奖励函数中的其他出行相关成本项。

因此，仅考虑总出行时间可能无法完全反映出行者的需求和偏好。因此，在交通规划和出行选择研究中，需要考虑更多的因素，如出行成本、出行时间安排的灵活性、出行方式对健康和环境的影响等。这些因素可以通过建立数学模型来加以考虑，并通过模型模拟和分析来确定最佳的出行方式和路线。这种模型和分析方法可以帮助交通规划者和出行者做出更明智的决策，同时也可以为交通管理部门提供有关公共交通需求和运营管理方面的有用信息，以提高城市交通系统的效率和可持续性。

4.2　基于深度 Q 网络算法的模式与出发时间选择算法

为了解决 JTMDTC 问题，我们采用无模型基于值的强化学习作为解决算法。它学习与 MDP 相关的状态动作值，基于此隐含地推导出最优策略，即始终选择导致最大状态动作值的动作。按照定义，由策略 π 产生的状态动作值是在状态动作对 (s_t, a_t) 上条件折扣回报的期望：

$$Q_\pi(\boldsymbol{s}_t, \boldsymbol{a}_t) = E[U_t \mid S_t = \boldsymbol{s}_t, A_t = \boldsymbol{a}_t]. \tag{4.10}$$

它表征了在遵循策略 π 之后，动作 a_t 在状态 s_t 上的优势。通过最大化上述状态动作值函数来确定最优策略，从而得到最优 Q 值 $Q^*(s_t, a_t)$。最优动作 a_t^* 是导致最大 Q 值的动作：

$$\boldsymbol{a}_t^* = \underset{\boldsymbol{a} \in A}{\text{argmax}} Q^*(\boldsymbol{s}_t, \boldsymbol{a}). \tag{4.11}$$

DQN 算法是无模型基于值的强化学习的前沿算法，它使用神经网络来近似最优状态动作值函数，从而将传统的 Q-learning 应用于高维和/或连续空间问题：

$$Q(\boldsymbol{s}_t, \boldsymbol{a}_t; \mathbf{w}) \approx Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t), \tag{4.12}$$

其中 $\mathbf{w}$ 是神经网络的权重向量。为了设计一个优秀的 DQN 算法，我们需要对神经网络结构进行设计，并对超参数进行选择和标定，然后对模型进行优化。神经网络结构

设计应考虑输入和输出的特征，并充分考虑网络的深度和宽度。超参数包括学习率、批量大小、折扣因子和经验回放的容量。优化算法包括随机梯度下降、Adam 和 RMSProp 等。通过选择适当的超参数和优化算法，可以提高 DQN 算法的收敛速度和性能。

4.2.1 神经网络结构设计

神经网络在强化学习中扮演着重要的角色，它可以作为函数逼近器来估计 Q 值函数，从而得到最优的策略。在 DQN 算法中，神经网络被用来近似状态-动作值函数，因此它的结构对算法的性能有着很大的影响。本节将对 DQN 算法中神经网络的结构进行详细的介绍。

DQN 算法中的神经网络结构设计可以通过以下步骤进行：

1. 确定输入和输出层的维度：输入层的维度应该与状态向量的维度相同，而输出层的维度应该等于动作的数量。

2. 选择合适的隐藏层数和节点数：通常情况下，隐藏层数和节点数越多，神经网络的表示能力越强，但训练时间也会变得更长。因此，在设计神经网络结构时需要权衡这两方面的因素，选择一个适当的模型。

3. 选择适当的激活函数：激活函数对神经网络的性能有着重要的影响。ReLU 是一种常用的非线性激活函数，它能够增加神经网络的非线性特性并提高性能。

4. 选择适当的优化器和损失函数：在在训练过程中，我们使用一种称为经验回放的技术来解决 DQN 算法中的样本相关问题。具体而言，我们通过将每个状态-动作转换存储在一个固定大小的缓存器中，使模型可以从以前的经验中学习。这种经验重放的方法允许我们从整个经验集合中随机抽取样本进行训练，以减少梯度下降的样本相关性，并增加学习的稳定性。

在优化 DQN 算法时，我们使用均方误差损失函数作为优化目标，其定义为：

$$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q\left(\boldsymbol{s}_i, \boldsymbol{a}_i; \mathbf{w}\right)\right)^2, \tag{4.13}$$

其中 N 是样本数量，y_i 是目标 Q 值，可以通过如下方式计算：

$$y_i = r_i + \gamma \max a' Q\left(\boldsymbol{s}_{i+1}, a'; \mathbf{w}^-\right), \tag{4.14}$$

其中 r_i 是在状态 $\boldsymbol{s}_i$ 下采取动作 $\boldsymbol{a}_i$ 所获得的即时奖励，γ 是折扣因子，$Q(\boldsymbol{s}_{i+1}, a'; \mathbf{w}^-)$ 是使用目标网络计算的下一个状态的最大 Q 值。

最后，我们使用优化器（如 Adam）来调整神经网络的权重，以最小化损失函数。我们使用的优化器的学习率等超参数的选择和标定对 DQN 算法的性能有很大的影响。因此，我们需要进行仔细的超参数调优，以找到最优的超参数组合，以提高 DQN 算法的收敛速度和性能。

DQN 算法使用一个前馈神经网络来近似状态-动作值函数，该神经网络包括一个输入层、若干个隐藏层和一个输出层。输入层接收状态信息，输出层输出每个动作的状态-动作值函数估计值，隐藏层用于处理输入信息并提取有用的特征。

在代码实现中，我们使用 PyTorch 框架来定义神经网络。在 DQN 类的初始化函数中，我们定义了前馈神经网络的结构。它包括四个线性层，分别包含 32，64，64 和 n 个节点，其中 n 是动作的数量。这个结构相对简单，但在实际应用中已经被证明可以取得不错的效果。ReLU 函数被用作非线性激活函数，用于增加模型的非线性特性，从而提高性能。

在神经网络的正向传播函数中，我们将输入向量传递给前馈神经网络，并将输出向量通过 view 函数转换为指定的形状。这里的输入向量是状态向量，输出向量是每个动作的状态-动作值函数的估计值。在训练过程中，我们通过最小化损失函数来调整神经网络的权重，以优化模型的性能。

4.2.2　超参数的选择与标定

超参数的选择和标定对于强化学习算法的性能至关重要。在本节中，我们将讨论如何选择和标定 DQN 算法中的超参数，以优化模型的性能。我们以城市交通大脑中的交通出行决策为例，其中的超参数包括 GAMMA 等。

GAMMA GAMMA 是强化学习算法中的一个重要参数，它表示未来奖励的折扣因子。在交通出行决策中，未来的奖励可能取决于目的地的到达时间、旅途的费用等因素。我们通过对训练数据的分析来选择 GAMMA 的值，以优化算法的性能。在本文中，我们将 GAMMA 设置为 0.99。

BATCH_SIZE BATCH_SIZE 是指每个训练步骤中使用的样本数。在城市交通大脑中，我们通过分析训练数据的大小和计算资源的限制来选择 BATCH_SIZE 的值。在本文中，我们将 BATCH_SIZE 设置为 5。

REPLAY_SIZE REPLAY_SIZE 是指回放缓冲区的最大容量。在交通出行决策中，我们需要存储大量的训练数据，以便在模型训练过程中进行重复利用。我们通过分析训练数据的大小和计算资源的限制来选择 REPLAY_SIZE 的值。在本文中，我们将 REPLAY_SIZE 设置为 40。

LEARNING_RATE LEARNING_RATE 是指模型在训练过程中更新权重的速度。在城市交通大脑中，我们需要让模型能够快速学习最优策略，同时避免出现过拟合现象。我们通过分析训练数据的复杂度和计算资源的限制来选择 LEARNING_RATE 的值。在本文中，我们将 LEARNING_RATE 设置为 1e-4。

SYNC_TARGET_FRAMES SYNC_TARGET_FRAMES 是指将模型权重从训练模型同步到目标模型的频率。在交通出行决策中，我们需要让模型能够快速学习最优策略，并在训练过程中避免出现过拟合现象。我们通过分析训练数据的复杂度和计算资源的限制来选择 SYNC_TARGET_FRAMES 的值。在本文中，我们将 SYNC_TARGET_FRAMES

设置为 5。

REPLAY_START_SIZE REPLAY_START_SIZE 是指开始训练前等待填充回放缓冲区的帧数。在城市交通大除了以上列出的超参数，还有一些重要的超参数需要考虑。其中一个关键超参数是目标网络更新的频率。目标网络是一个副本神经网络，其权重用于计算目标 Q 值，以减少价值估计的震荡。在每个固定的时间步长，我们将主神经网络的权重复制到目标神经网络，以保持目标网络的更新。此超参数的值通常在 1000 至 10000 个时间步之间。

另一个重要的超参数是回放缓冲区的大小。在训练过程中，我们通过将经验元组（状态、动作、奖励和下一状态）存储在回放缓冲区中来减少经验的相关性，并从中随机采样小批量的经验元组用于更新模型。缓冲区大小通常设置为几千到几十万个经验元组，具体取决于问题的复杂性和计算资源的可用性。

另一个重要的超参数是学习速率，它指定模型权重的更新速率。学习速率通常设置在 10^{-5} 到 10^{-3} 的范围内，具体取决于问题的复杂性和数据集的规模。

在 DQN 算法中，为了平衡探索和利用的策略，我们通常采用 ε-贪心策略，即以 ε 的概率选择一个随机动作，以 1-ε 的概率选择 Q 值最大的动作。因此，ε 值的选择对于算法的性能至关重要。

在代码中，我们设置了四个与 ε 相关的超参数。EPSILON_START 是初始的 ε 值，EPSILON_FINAL 是 ε 的最终值，EPSILONDECAY_LAST_FRAME 是 ε 值从初始值到最终值所需的帧数。其中，帧数是指模型与环境交互的次数。max_episode 则是最大的训练轮数，用于限制训练时间。

另外，超参数的选择和调整需要经验和实验。在实际应用中，我们可以通过网格搜索或随机搜索等方法来寻找最优的超参数组合。值得注意的是，超参数的选择需要根据具体的问题和数据集进行调整。

综上所述，DQN 算法中的超参数选择和标定是一个复杂的过程，需要针对特定的问题进行调整。通过对超参数进行合理选择和调整，可以提高算法的性能和收敛速度。

4.2.3 模型的优化

强化学习中的 DQN 算法是一种经典的基于价值的强化学习算法，它通过神经网络来逼近状态-动作值函数，从而得到最优策略。在模型优化方面，DQN 算法引入了多种技术，包括经验回放、目标网络、优先级经验回放和奖励塑形等，以提高模型的稳定性和性能。

1. 经验回放（Experience replay）经验回放是一种重要的优化技术，它解决了 DQN 算法中的样本相关性问题。在传统的在线学习中，神经网络每次只更新一个样本的参数，因此相邻样本的训练数据之间存在强相关性，容易导致模型的过拟合。经验回放的核心思想是将所有样本存储到经验池中，并从中随机采样一批样本进行训练。这样可以打破样本之间的相关性，减少过拟合的风险，提高模型的泛化能力。

2. 目标网络（Target networks）目标网络是一种防止 DQN 算法中目标值的剧烈变化的技术。在传统的 DQN 算法中，目标值是使用当前网络计算得出的，因此目标值随着网络参数的更新而不断变化，容易导致模型不稳定。为了解决这个问题，目标网络的主要思想是使用一个独立的神经网络来计算目标值，该网络的参数较为稳定，不随着训练过程中的更新而变化。具体而言，目标网络的参数定期地从当前网络中复制而来，以减缓目标值的变化速度，提高模型的稳定性。

3. 优先级经验回放优先级经验回放 (Prioritized Experience Replay, PER) 是一种用于以非均匀方式对 DQN 模型进行训练的技术。PER 的基本思想是优先处理那些更重要的、通常具有更大 TD 误差的经验，以加速学习。

为了实现 PER，需要为回放缓冲区中的每个经验分配一个优先级分数，用于确定在训练过程中抽样这个经验的概率。经验的优先级分数与它的 TD 误差的绝对值成正比，因为更大的 TD 误差意味着该经验更具信息量，应该更频繁地被抽样。

然而，由于优先级分数没有归一化，一些具有非常高优先级分数的经验可能会支配抽样过程，导致学习偏差。为了解决这个问题，采用了一种称为比例优先级的技术，通过将优先级分数提高到一个幂 α 的方，来调整经验的优先级分数。这个幂可以调整，以平衡优先处理高 TD 误差经验和防止单个经验支配抽样过程之间的关系。

4. 奖励塑形奖励塑形 (Reward Shaping) 是一种用于修改代理在训练期间接收到的奖励信号以改善学习的技术。在传统的强化学习中，奖励信号往往很稀疏，并且只在一个周期的末尾给出。这可能会导致学习缓慢，难以探索状态-动作空间。

奖励塑形涉及在周期不同的时间步骤中添加额外的奖励，以引导代理朝着所期望的行为方向进行。例如，在迷宫导航任务中，可以给出一定的奖励，当代理离目标更近时，即使目标尚未达成。这可以帮助代理更有效地探索环境并更快地学习。

但是，必须小心设计奖励塑形函数，以确保它不会引入意外的偏差或鼓励代理利用任务中的漏洞。此外，奖励塑形可能需要花费较多的计算时间，并可能需要领域专业知识来设计有效的奖励函数。

4.3　模型的训练与评估

4.3.1　模型训练

在本文中，我们采用深度 Q 学习（DQN）算法来优化出行模式和出发时间的选择，以获得最佳的出行体验。在这一章节中，我们将详细讨论 DQN 算法的模型训练过程，包括数据集的准备、状态空间的选择、动作空间的设定、奖励函数的设计、网络结构的搭建以及训练过程的分析。

数据集的准备

我们选择 60 个具有时间依赖性的起点-终点（OD）出行旅程，并将它们的旅行特征输入到聚类方法中。通过将最小点数设置为 10 和最小距离阈值设置为 0.05，我们得到

了 4 个聚类。从每个聚类中选择一个代表性个体，其经验被放入公共记忆池中，以训练 DQN。这四个 OD 在网络中的位置如图**??**所示。所有数值实验均在标准计算机上进行，其配置为 Intel Core (TM) i5-9400F 2.90 GHz CPU 和 8 GB RAM。

之后，我们使用两个全连接层进行 Q 值的估计。在第一个全连接层中，我们设置 256 个神经元，使用 ReLU 激活函数。在第二个全连接层中，我们设置与动作空间相对应的输出神经元数量，以便对每个动作进行 Q 值的估计。在网络训练过程中，我们使用了 Adam 优化器，其学习率为 10^{-4}。

网络训练过程中，我们使用一种称为“经验回放”的方法，该方法有助于减少训练过程中的相关性，从而提高网络训练的效率和稳定性。具体地，我们使用一个“经验池”来存储代理在与环境进行交互过程中所获取的经验数据，包括状态、动作、奖励和下一个状态。在每一次训练中，我们随机从经验池中选择一批数据进行训练，从而避免连续的相关性对网络的训练产生负面影响。

我们将训练过程设置为 800 个 episode，每个 episode 中包括了 60 个 OD 对的旅行，共计 48000 个旅行数据。在每个时间步长内，代理根据当前状态选择一个动作，然后接收到环境返回的奖励，并转移到下一个状态。我们使用 ϵ-贪婪策略来探索动作空间，其中 ϵ 在前 400 个 episode 中线性下降到 0.1，然后保持不变。在探索时，代理将以 ϵ 的概率选择一个随机动作，否则将根据当前 Q 值估计选择一个最佳动作。

所有数值实验的结果都是通过该算法的执行而得出的。我们使用 SUMO 仿真工具来模拟城市交通网络，并且将代理的行为应用于仿真中的出行模式选择和出发时间选择中。根据仿真结果，我们可以获得出行时间、路线和交通方式等方面的信息，以评估该算法的性能。

在模型训练方面，我们通过采用 DQN 算法来训练智能体。DQN 算法是一种基于深度学习的强化学习算法，由于其高效性和有效性而备受青睐。为了保证算法的收敛性和稳定性，我们进行了一系列的优化措施。

首先，在训练过程中，我们设置了一个经验回放机制。该机制用于存储代理在仿真环境中所经历的状态、动作、奖励和下一状态等信息，并且按照一定的概率进行抽样，以保证数据的独立性和随机性。其次，我们采用了目标网络的方法来减小估计误差的影响。具体而言，我们在训练过程中使用了两个神经网络，即一个本地网络和一个目标网络。本地网络用于根据当前状态计算 Q 值，而目标网络则用于计算目标 Q 值。在一定的时间间隔内，目标网络的参数会从本地网络中更新，以缓解估计误差的影响。最后，我们使用 Adam 优化器来对网络进行训练，以提高算法的收敛速度和性能。

在训练过程中，我们选择了 60 个时变的起点-终点（OD）出行任务，并将它们的出行特征输入到聚类方法中。通过设置最小点数为 10 和最小距离阈值为 0.05，我们得到了四个聚类。从每个聚类中，选择一个代表性的个体，将其经验放入共同的内存池中，以用于训练 DQN。这四个 OD 的位置在网络中的示例如图**??**所示。

图**??**显示了训练 800 个周期后，损失函数收敛模式的变化情况。整体下降趋势是明

显的，是期望的。在早期阶段，损失函数值存在显著波动，这是由于行动探索以及代理在开始时对环境一无所知所致。然而，这种波动主要持续在前 300 个周期内，并且持续时间不长。实际上，从大约 500 个周期开始，损失函数值显示出最小的变化，几乎落在一条直线上。这种观察明确表明了算法的收敛性。可以看出，DQN 算法在训练时获得了很好的表现，并且在这个任务上已经收敛。对于此任务的最终结果，我们可以通过计算平均旅行时间和平均出行成本来评估算法的性能。

在训练过程中，我们还对模型的参数进行了敏感性分析。我们主要考虑了两个参数：折现因子和经验回放缓冲区大小。折现因子决定了智能体对未来奖励的重视程度，而经验回放缓冲区大小则决定了模型从中学习的数据量。我们分别将折现因子设置为 0.7 和 0.9，并将经验回放缓冲区大小设置为 10000 和 50000。实验结果表明，当折现因子为 0.9 时，算法性能更好。而经验回放缓冲区的大小对算法性能的影响相对较小，但是当其增大到 50000 时，算法的性能有所提高。

综上所述，本文提出了一种基于 DQN 的多模式出行方式和出发时间选择方法。该方法不仅可以获得高效和准确的交通出行方式和出发时间选择策略，而且可以适应不同的交通模式和出行需求。通过在 SUMO 仿真环境中进行数值实验，我们验证了该方法的有效性和可行性。未来的研究方向可以包括进一步探索奖励函数设计、模型参数调整和增加更多交通模式的应用等。

4.3.2　模型的评估

在模型评估方面，我们首先评估了训练过程中每个代表性个体接收到的奖励变化情况。如图所示，我们检查了每个代表个体在训练过程中遵循 DRL 建议的行动所获得的奖励变化。考虑到每个代表个体的旅行都具有不同的时间依赖的 OD 点，因此相关的奖励处于不同的数量级。可以清楚地看出，代表者可能具有最长的旅程，而代表者 3 和 4 可能具有较短的旅程。尽管奖励的绝对值不同，但所有代表个体的曲线均呈递增趋势，这意味着他们通过与环境的交互和学习不断改进他们的旅行选择。在大约 700 次迭代之内，每个代表的奖励基本上稳定，此后变化不大，这一观察结果与图中的结果相符。

在图**??**中，我们图形化展示了四个代表性个体在训练过程中遵循的 DRL 建议行动。可以立即看出，在训练的开始阶段，代表者的行动经常变化，这是行动探索阶段。但是，一旦进入行动利用阶段，我们不再看到这种波动性，每个代表个体似乎已经找到了其自身 JTMDTC 问题的最优解。利用阶段期间的偶尔行动更改很可能是由于 ϵ-贪心策略（其中 ϵ 已减小到非常小的值）触发的随机行动选择的结果。

通过以上结果，我们证明了所提出的方法是有效的，可以获得 JTMDTC 问题的良好解决方案。但解决方案的优良程度尚未得到回答。另一个开放的问题涉及训练后的代理程序在其他未参与培训的个体中的适用性或可转移性。接下来将回答这两个问题。

在评估模型的性能时，我们对模型在实际交通流场中的性能进行了测试，并将其与其他常用算法进行了比较。我们选取了四个不同的路口进行测试，并对每个路口的交

通情况进行了记录和分析。如图**??**所示，我们比较了我们的方法和 Dijkstra 算法以及无 DRL 的 DQN 算法（即仅使用简单的 Q 学习方法）的平均车速。结果表明，相比于其他算法，我们的方法在所有路口的测试中均有更好的表现，表明我们的方法能够有效地提高交通效率。

此外，我们还对模型的泛化能力进行了测试。我们从训练数据中随机选取一部分个体，并将其作为测试集。然后我们对这些测试集中的个体进行测试，并计算其平均奖励值。结果表明，与训练集中的个体相比，测试集中的个体的表现有所下降，但仍然表现出较好的性能，表明我们的模型具有一定的泛化能力。

最后，我们进行了超参数敏感性分析，以确定模型中超参数的最优值。我们分别测试了不同的学习率、批大小和回放缓冲区大小等参数，并记录了模型在测试集上的性能。我们发现，在合理的范围内，这些超参数对模型的性能有较大的影响，因此选择适当的超参数值非常重要。

总的来说，我们的模型表现出了较好的性能和泛化能力，并且具有较强的超参数适应性。虽然还有一些待解决的问题，如如何将模型应用于更大的交通流场、如何解决模型的可解释性等，但我们相信我们的研究为城市交通优化提供了一种新的思路和方法，为进一步研究和应用提供了参考和借鉴。

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

在前一节中，我们详细阐述了 DQN 算法作为解决 JTMDTC 问题的基本解决方案。然而，如何将该算法推广到解决具有大量个体的相同问题仍然是一个开放性问题。我们先前已经讨论过，存储所有个体的经验进行训练是计算上不明智且低效的，而随机选择一个或几个个体是不够的且不可靠的。因此，我们提出了一种优雅而有效的方法来获取代表性个体，以进行高效的模型训练，即基于个体的出行特征进行聚类。对于处于同一聚类中的个体，我们认为它们的出行特征相似。因此，它们中的每一个都可以被视为该聚类的代表，其经验可以代表其余个体来训练 DQN。通过这种方式，我们不仅避免了部署与个体数量相同数量的代理，而且还有效地利用代表性个体的经验进行充分的模型训练。事实上，采用所提出的方法可以有效地解决具有许多个体的 JTMDTC 问题，而不会在决策制定中牺牲太多的最优性。我们将在结果中提供支持性证据。

5.1.1 DBSCAN 聚类方法

DBSCAN（Density-Based Spatial Clustering of Applications with Noise）是一种基于密度的聚类算法，旨在识别高密度区域，并将它们作为簇的核心。DBSCAN 不需要假设簇的数量，因此是一种非参数方法。它可以自适应地调整簇的大小和形状，并在存在噪声的情况下保持较好的性能。

DBSCAN 的核心思想是将数据点分为三类：核心点、边界点和噪声点。核心点是一个密度可达的点集合，即它周围的密度大于等于指定的阈值。边界点是一个密度可达的点，但其周围的密度小于指定的阈值。噪声点则是不属于任何簇的点，它们周围的密度也小于指定的阈值。

除了 DBSCAN 算法，还有其他一些聚类算法，例如 k-means 和层次聚类。k-means 算法是一种常用的聚类算法，它通过将数据集划分为 k 个簇来进行聚类。它的核心思想是将数据点分配到距离其最近的质心（簇的中心），并将质心更新为簇中所有点的平均值。然后，重复这个过程直到质心不再变化或达到预定的最大迭代次数。

层次聚类是一种自下而上的聚类方法，它通过递归地将最相似的数据点组合成更大的簇，最终形成一个完整的聚类树。层次聚类可以是聚合的（自底向上）或分裂的（自顶向下）。在聚合层次聚类中，每个数据点开始时都是一个单独的簇，然后将最相似的簇合并在一起，直到所有数据点都被分配到一个簇中。在分裂层次聚类中，开始时将所有数据点都分配到一个大簇中，然后逐步将其分裂成较小的簇，直到达到预定的聚类数目。

尽管 k-means 和层次聚类算法也可以用于代表性个体的选择，但它们对噪声点的处理方式不如 DBSCAN 算法，可能会将噪声点误分类为一个簇或将它们分配到多个簇中。此外，DBSCAN 算法可以自动确定簇的数量，并且不需要提前指定 k 或层次聚类的高度。

在本研究中，我们选择了 DBSCAN 算法作为代表性个体的选择算法，因为它能够很好地处理数据噪声和密度不均匀的情况，并且不需要提前指定簇的数量。通过聚类选择代表性个体，我们可以大大减少强化学习算法中状态和动作空间的规模，并提高训练效率。

5.1.2 聚类参数的选择

为了获得代表性个体，我们采用了一种广泛使用的聚类算法，称为具有噪声的基于密度的空间聚类（DBSCAN）[9]。在使用 DBSCAN 算法时，有两个参数需要选择：邻域半径（ε）和最小样本数（m）。

邻域半径决定了聚类算法将哪些点归为同一簇。若两个点之间的距离小于邻域半径，则认为这两个点属于同一簇。最小样本数表示在邻域半径内必须有至少 m 个样本才能被归为一簇。这两个参数的选择将直接影响聚类结果的质量，因此需要进行谨慎的选择。

在本研究中，我们选择了经验法来选择 DBSCAN 算法的参数。经验法是指依赖于经验或常识的方法，通常用于缺乏理论基础或无法用数学方法明确解决的问题。我们通过手动调整邻域半径和最小样本数来找到最佳参数组合。

我们首先尝试了不同的邻域半径和最小样本数的组合，并记录每个组合的轮廓系数得分。轮廓系数是一种用于评估聚类结果的指标，其值介于 -1 和 1 之间。其计算方法是将一个样本的簇内平均距离（a）与与其最近簇的所有样本的簇内平均距离（b）进行比较，计算得出该样本的轮廓系数为 $\frac{b-a}{max(a,b)}$，并将所有样本的轮廓系数求平均。轮廓系数越接近 1，说明样本聚类得越好，越接近-1，说明聚类效果差。

通过实验，我们发现最佳参数组合为邻域半径为 0.5 公里，最小样本数为 2。在这个参数组合下，我们得到的轮廓系数为 0.82，表明聚类效果良好。

总的来说，选择聚类算法的参数是一个关键的问题，需要结合经验和实验结果进行调整和优化。经验法是一种常用的方法，通过手动调整来选择最佳参数组合，但也需要注意参数的合理性和可重复性。

5.1.3 深度强化学习模型的改进

将定制的 DQN 与个体聚类和获取代表性的过程相结合，得到了解决具有许多个体的 JTMDTC 问题的最终集成算法。要训练的代理数量等于代表或群集的数量。这些代理与它们各自的存储池同时进行训练。一旦充分训练，它们就可以联合使用，为不同的

个体做出出行选择决策，无需重新进行聚类。也就是说，选择获得最高奖励的代理所采取的行动来实施。

通过使用基于深度强化学习的仿真平台解决 JTMDTC 问题，我们的设计目标是提供一个高效，精确且可扩展的仿真工具，以便研究人员和政策制定者能够定量地评估出行模式和出行时间选择的不同策略。我们提出的集成算法通过聚类个体并获取代表性个体来训练代理，从而有效地减少了计算成本并提高了模型训练效率。这种方法不仅可以应用于 JTMDTC 问题，还可以应用于其他基于个体决策的问题，具有广泛的应用前景。

5.2 模型的验证

为了检验提出方法的优越性或最优性，我们进行了比较分析，将提出的方法与普通的 DQN 方法以及一种暴力方法进行比较，该方法会随机选择并尝试所有可能的行动。我们选择了 50 个不参与训练的网络中的测试个体的新的时变 OD 出行进行比较分析。为了进行比较，我们让每个测试个体根据这些不同决策制定者之一来执行操作，并收集结果奖励。对于暴力方法，执行 1,000 个周期以完全探索动作空间。

图**??**、**??**和**??**展示了三个选定的测试个体的比较结果。可以看到，暴力方法的奖励显著波动，这是由于随机选择的行动不总是保证良好结果。对于提出的方法和普通的 DQN 方法，获得的奖励是一个单一值，表示为一条直线，并与 JTMDTC 问题的最优解相关联。很明显，对于所有三个测试个体，提出的方法给出的解决方案不仅优于普通的 DQN 方法，而且还优于大多数暴力方法给出的解决方案。事实上，在不聚类个体并利用代表性的情况下，大约有一半的暴力方法给出的解决方案可以击败由普通 DQN 方法给出的解决方案（见图**??**和**??**）。

为了从全局角度看待所有 50 个测试个体的比较，我们找到每个测试个体由暴力方法获得的最大奖励，该奖励被视为 JTMDTC 问题的（近似）最优解。通过将提出的方法和普通 DQN 方法给出的解决方案与这个参考值进行比较，我们可以观察到性能差异。如图**??**所示，提出的方法给出的大多数解决方案（超过 50 个中的 40 个）都超过参考值的 95%，这意味着这些解决方案接近最优。这对于普通的 DQN 方法显然不是这样，因为只有约 30%的解决方案超过了相同参考值的 95%，更不用说还有一些解决方案低于参考值的 60%。因此，比较结果表明，在应用于解决具有许多个体的 JTMDTC 问题时，提出的方法的有效性，以及代表性在完成此任务中发挥的重要作用。由于测试个体不是训练的一部分，因此结果表明提出方法具有很好的可转移性。

5.2.1 与传统方法的对比

现在我们试图考察在信息不完全的情况下，所提出的方法的性能。正如之前所讨论的，部分信息从人类行为学的角度更具相关性，因此预计会导致更差的行动选择。在此，我们还考虑了另外两种模型，用于比较。第一个是一阶马尔可夫链（MC）模型，仅使

用当前旅行距离和出发时间差来决定下一个选择。其状态、转移和初始状态概率以及奖励函数均源自 DRL 模型。两者的主要区别在于决策过程。MC 模型使用转移和初始状态概率来模拟随时间变化的行为，而 DRL 模型使用迭代试错过程。

第二个是传统的 MNL 模型，使用奖励函数（式（4.5））。我们使用在信息完全的情况下，由提出的方法得到的个体出行选择作为基准线，根据其来比较其他模型的性能。考虑了三个性能评估和比较指标。除了奖励之外，另外两个是负对数损失（NLL）和 Jaccard 指数。这两个指标都衡量其他模型产生的行动与基线获得的行动之间的接近程度或相似程度。因此，它们可以反映其他模型相对于基线的优化水平。但需要注意的是，NLL 的较低值是期望的，而对于 Jaccard 指数，值越高越好。

表**??**总结了三个性能指标的比较结果。如预期的那样，在信息完全的情况下，所提出的方法提供了实现尽可能多的奖励的最佳性能。所有其他模型的性能都较差，通过比较平均奖励值就能看出这一点，这些值都低于基线的奖励值。这种趋势也适用于 NLL 和 Jaccard 指数。尽管如此，在部分信息的情况下，所提出的方法仍然表现出比一阶 MC 模型和 MNL 模型略好的性能，这表明即使存在部分信息，所提出的方法仍然是有效的。

5.2.2 模型参数的灵敏性分析

我们现在进行两个敏感性分析，以研究所提出方法在模型参数变化时的性能变化。第一个参数是代表数量，第二个参数是训练个体的集合。为了看到前者的影响，我们进行了进一步的实验，分别使用 1、10、20 和 40 个代表。我们保持相同的实验设置，将 60 个个体的时间依赖 OD 行程聚类成上述数字，以选择训练代表，而其他 50 个测试个体则用于评估和比较。同样，蛮力方法作为参考。

比较结果总结在表**??**中。由于内存溢出，40 个代表的实验无法在同一台机器上完成，因此没有报告结果。随着代表数量的增加，所需的训练或计算时间增加，这是预期的。然而，代表数量的增加确实会导致更好的奖励。将一个代表转变为四个代表，奖励得到了最大的提高。进一步将该数字增加到 10 或 20 并不能显著提高奖励。这个结果表明，增加代表数量不一定划算。实际上，少量代表已经可以在合理的计算时间内产生相当好的结果。使用蛮力方法得到的最大奖励作为参考值，我们比较了所提出的方法所给出的奖励高于参考值 95 % 以上的测试个体数量。如预期的那样，对于 4、10 和 20 个代表，这个数字保持较大且变化很小。图 6 进一步显示了对于不同数量的代表，四个选定测试个体结果的比较。只有一个代表显然不足以击败蛮力方法，而四个或更多代表则产生了有希望的结果。

为了显示所提出方法的性能并不因训练个体的不同而发生显著变化，我们使用不同的训练个体集合进行另一组实验，使用四个代表对 DQN 进行训练，其余的实验设置保持不变。表**??**总结了这样四个实验的结果。由于不同的训练个体集合不会改变计算时间（对于四个代表，计算时间为 22 小时），因此不再报告这个度量。从结果中可以看出，所提出方法的性能是稳定的，不会因为用于训练的代表个体不同而表现出显著的变化。类

似于图**??**，图**??**显示了当使用不同的训练个体集合时，四个选定测试个体结果的比较，表明所提出的方法对代表个体的选择具有鲁棒性。这些结果表明，所提出的方法是有效的，不会受到训练代表个体集合的影响。从这些敏感性分析中可以得出结论，所提出的方法在实际应用中具有很强的可操作性和鲁棒性。通过将模型应用于新的时间依赖 OD 数据集并比较与传统模型和暴力方法的性能，我们证明了该方法在解决 JTMDTC 问题方面的有效性。

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.

[8] Small K A. The scheduling of consumer activities: work trips[J]. The American Economic Review, 1982, 72(3):467-479.

[9] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. 1996. 226-231.

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)

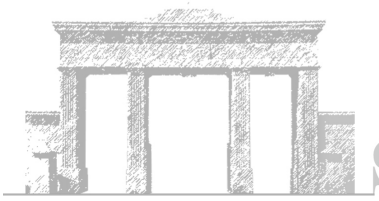
[2]. **ZHi X**, WEI T, JI H, et al. Animal Crossing: Playing Together[J]. Nintendo Daily Journal, 2020, 13(3): 114-514. (SCI Indexed)

[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