Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Управління транспортними потоками на основі машинного навчання

Текстова частина до магістерської роботи за спеціальністю «Інженерія програмного забезпечення»

Виконав студент 2-го року навчання
Велігурський Олександр Сергійович
Керівник
Франчук О. В. доцент, к.н.

Магістерська робота захищена з
оцінкою «_____»

Секретар ДЕК _____
«___»_____ 2023 р.

Київ 2023

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
доц., канд. наук
_____ О. В. Франчук
(підпис)
1 листопада 2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на магістерську роботу

студенту Велігурському О. С. факультету інформатики 2-го курсу МП
Тема: «Управління транспортними потоками на основі машинного навчання»

Зміст ТЧ до дипломної роботи:

Індивідуальне завдання
Вступ
1. Дослідження наявних методів
2. Вибір та розробка алгоритмів та системи управління
3. Аналіз розробленої моделі та порівняння з ручним керуванням

Висновки
Список використаних джерел

Дата видачі 1 листопада 2022 р.

Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Календарний план виконання роботи:

№ п/п	Назва етапу дипломної роботи	Термін виконання	Примітки
1	Отримання завдання на дипломну роботу	01.11.22	
2	Огляд теоретичного матеріалу за темою роботи та аналіз існуючих рішень	20.01.23	
3	Опис теоретичної частини	25.02.23	
4	Опис вимог до практичної частини	01.03.23	
5	Реалізація прототипу програмного застосунку	01.04.23	
6	Написання пояснювальної записки	15.04.23	
7	Аналіз отриманих результатів, написання доповіді та попередній захист магістерської роботи	18.05.23	
8	Корегування роботи за результатами попереднього захисту	20.05.23	
9	Створення слайдів для доповіді та написання доповіді	25.05.23	
10	Захист	14.06.23	

Анотація

Дипломна робота присвячена дослідженню можливості використання методів машинного навчання для управління транспортними потоками. Метою роботи є розробка алгоритму для передбачення кількості транспорту на ділянці дороги в реальному часі та прийняття рішень щодо управління транспортними потоками. Включає розробку програмного забезпечення для моделювання та експериментального дослідження розробленої моделі, а також проведення експериментів та аналіз результатів дослідження.

Перший розділ містить дослідження наявних методів машинного навчання та нейронних мереж для їх використання в управлінні транспортними потоками. Описуються принципи роботи, методи тренування моделей.

У другому розділі проводиться вибір кращого підходу для отримання позитивних результатів. Описуються обрані технології та алгоритми, кроки які були зроблені для навчання моделі та опис створення середовища для тестування розробленого алгоритму.

У третьому розділі роботи розглядаються отримані результати та реалізація алгоритмів для управління транспортними потоками на основі машинного навчання. Описуються порівняння отриманих результатів з ручним керуванням трафіку. Проводиться аналіз інших можливостей для покращення розробленої системи.

Вступ

З швидким розвитком міст та збільшенням чисельності населення, управління транспортними потоками стає все більш актуальним завданням. Оскільки змінити інфраструктуру міста, додати нові дороги або збільшити існуючі дуже часто є неможливим, то потреба в ефективному управлінні транспортними потоками стає критично важливою для забезпечення міської мобільності та зменшення транспортних заторів.

Одним з підходів до розв'язання проблеми кількості транспорту на дорогах є використання машинного навчання. Машинне навчання може допомогти в розв'язанні проблем, пов'язаних з прогнозуванням та управлінням транспортними потоками, шляхом аналізу великих обсягів даних та використання різних алгоритмів навчання.

Метою даної дипломної роботи є розробка та дослідження моделі управління транспортними потоками на основі машинного навчання. Робота включає в себе вибір та розробку алгоритмів машинного навчання, розробку програмного забезпечення для моделювання та експериментального дослідження розробленої моделі, а також проведення експериментів та аналіз результатів дослідження.

Отже, результати даної дипломної роботи можуть бути корисними для розробки більш ефективних та точних систем управління транспортними потоками в містах.

1 Наявні проблеми в управлінні транспортними потоками

Управління транспортними потоками є однією з ключових задач у галузі транспортної інженерії та міського планування, оскільки воно має вирішальний вплив на безпеку, комфорт та ефективність транспортних систем. За допомогою технологій машинного навчання можна покращити управління транспортними потоками, зменшити затори та покращити ефективність дорожнього руху.

Однією з проблем, що виникає в процесі управління транспортними потоками, є прогнозування трафіку. Для цього застосовуються різні методи машинного навчання, такі як нейронні мережі, глибинне навчання, дерева рішень та інші. Вони допомагають прогнозувати рух транспорту на основі різних факторів, таких як погода або час доби. Ще одна проблема, що потребує уваги, це оптимізація маршрутів руху транспорту. Тут також можуть застосовуватися методи машинного навчання, які допоможуть врахувати різні фактори, такі як затори на дорозі, стан дорожнього покриття, кількість автомобілів тощо.

Наслідки недостатньої уваги до управління транспортними потоками можуть бути серйозними, такі як затори на дорогах, аварії, затримки пасажирів, забруднення навколишнього середовища та інші. Тому розгляд проблематики та дослідження наявних методів управління транспортними потоками на основі машинного навчання може бути корисним для покращення ситуації в цій галузі.

1.1 Машинне навчання

Машинне навчання (Machine learning) - це підгалузь штучного інтелекту в галузі інформатики, яка застосовує методи, алгоритми та статистичні моделі для надання комп'ютерам здатності «навчатися» (тобто, поступово покращувати продуктивність у певній задачі) з даних, без того, щоби бути програмованими явно.[1]

До основних методів машинного навчання належать:

1) Навчання з учителем (supervised learning): це один з основних методів машинного навчання, в якому модель навчається на основі вхідних даних та супутніх відповідей ("вчителем"), які передбачаються для кожного вхідного прикладу. Метою навчання з учителем є побудова моделі, яка може зробити приблизно точні передбачення для нових, раніше невідомих даних.

Навчання з учителем зазвичай використовується для задач класифікації та регресії. У задачі класифікації модель намагається призначити вхідному прикладу одну з певної кількості можливих міток або категорій (наприклад, визначити, чи є на зображенні автомобіль). У задачі регресії модель намагається знайти залежність між вхідними даними та вихідним значенням, яке може бути числом або вектором (наприклад, передбачити ціну на автомобіль на основі його характеристик).

2) Навчання з підкріпленням (reinforcement learning): це галузь машинного навчання, що вивчає питання про те, які дії повинні виконувати агенти в певному середовищі задля максимізації деякого уявлення про сукупну винагороду. В машинному навчанні середовище зазвичай розглядається як марковський процес вирішування , оскільки багато алгоритмів навчання з підкріпленням для цього контексту використовують методики динамічного програмування. Основна відмінність між класичними методиками й алгоритмами навчання з підкріпленням полягає в тому, що останні не потребують знання про МПВ, і вони орієнтовані на великі МПВ, в яких точні методи стають нездійсненними. Навчання з підкріпленням відрізняється від стандартного навчання з учителем тим, що пари правильних входів/виходів ніколи не представляються, а недостатньо оптимальні дії явно не виправляються. Крім того, є акцент на інтерактивну продуктивність, яка включає знаходження балансу між дослідженням та використанням. [2]

Один з прикладів використання навчання з підкріпленням в управлінні транспортними потоками - це оптимізація світлофорів на перехрестях. Модель навчається приймати рішення про те, які сигнали світлофора передавати, щоб максимізувати пропускну здатність дорожньої мережі та зменшити затори.

3) Навчання без учителя (unsupervised learning): це метод машинного навчання, при якому модель навчається без використання заздалегідь підготовлених даних з мітками або правильними відповідями. Модель повинна самостійно знайти внутрішні залежності в даних і створити корисну репрезентацію цих даних. Таким чином, навчання без учителя зазвичай використовується для знаходження певних структур або патернів в даних.

У контексті управління транспортними потоками, навчання без учителя може бути використане для аналізу руху транспорту на основі відеоданих з камер спостереження або з датчиків руху, з метою виявлення патернів руху автомобілів та визначення характеристик дороги, що впливають на рух транспорту. Отримана інформація може бути використана для покращення управління транспортними потоками та зменшення заторів.

1.2 Нейронні мережі

Нейронні мережі також відомі як штучні нейронні мережі (ANN) або імітовані нейронні мережі (SNN), є підмножиною машинного навчання та є основою алгоритмів глибокого навчання. Їх назва та структура навіяні людським мозком, імітуючи спосіб, яким біологічні нейрони передають сигнали один одному. Штучні нейронні мережі складаються з вузлових шарів, що містять вхідний рівень, один або більше прихованих шарів і вихідний рівень. Кожен вузол, або штучний нейрон, з’єднується з іншим і має відповідну вагу та поріг. Якщо вихід будь-якого окремого вузла перевищує вказане порогове значення, цей вузол активується,

надсилаючи дані на наступний рівень мережі. В іншому випадку дані не передаються на наступний рівень мережі.

Нейронні мережі покладаються на навчальні дані, щоб підвищувати свою точність з часом. Однак, коли ці алгоритми навчання налаштовані на точність, вони стають потужними інструментами в інформатиці та штучному інтелекті, що дозволяє класифікувати та кластеризувати дані з високою швидкістю. Завдання з розпізнавання мовлення або розпізнавання зображень можуть тривати хвилини чи години порівняно з ручною ідентифікацією експертів-людей. Однією з найвідоміших нейронних мереж є пошуковий алгоритм Google.[3]

Нейронні мережі використовуються для розв'язання складних задач, таких як обробка зображень, розпізнавання образів, прогнозування та інших завдань, для яких не можна легко визначити аналітичну формулу. У контексті управління транспортними потоками, нейронні мережі можуть бути використані для обробки зображень з камер відео нагляду за дорожнім рухом, а саме розпізнавати на зображенні різні типи транспорту, їх кількість, напрямок руху та аналізувати трафік на дорогах.

1.3 Розпізнавання рухомих об’єктів

Розпізнавання рухомих об'єктів камерами дорожнього руху - це один з методів, які можуть бути використані для управління транспортними потоками. Цей метод полягає у встановленні камер на дорогах, які здатні зафіксувати рух транспортних засобів та інших рухомих об'єктів на дорозі. Наступним кроком є обробка отриманих даних з використанням нейронних мереж. Для розпізнавання рухомих об'єктів камерами дорожнього руху можна використовувати нейронні мережі, які навчаються на великих масивах вхідних даних з камер. Ці мережі здатні розпізнавати транспортні засоби, пішоходів та інші рухомі об'єкти на дорозі, а також відрізняти їх від статичних об'єктів, таких як будівлі та дерева. Отримані дані

можуть бути використані для регулювання світлофорів, розрахунку оптимального швидкості руху транспорту, прогнозування трафіку та інших задач, пов'язаних з управлінням транспортними потоками. До переваг цього методу відносяться його точність, можливість обробки великих масивів даних та здатність до автоматизації процесу розпізнавання.

Однією з найпоширеніших технологій розпізнавання образів є OpenCV (Open Source Computer Vision Library) - бібліотека комп’ютерного зору та машинного навчання з відкритим кодом. OpenCV було створено, щоб забезпечити загальну інфраструктуру для програм комп’ютерного зору та прискорити використання машинного сприйняття в комерційних продуктах. Будучи ліцензованим продуктом Apache 2, OpenCV дозволяє компаніям легко використовувати та змінювати код.

Бібліотека містить понад 2500 оптимізованих алгоритмів, що включає повний набір як класичних, так і найсучасніших алгоритмів комп’ютерного зору та машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання облич, ідентифікації об’єктів, класифікації дій людей у відео, відстеження рухів камери, відстеження рухомих об’єктів, вилучення 3D-моделей об’єктів, створення 3D-хмар точок із стереокамер, з’єднання зображень для отримання високої роздільної здатності тощо. OpenCV має понад 47 тисяч користувачів спільнота та оціночна кількість завантажень перевищує 18 мільйонів. Бібліотека широко використовується компаніями, дослідницькими групами та державними органами. [4]

Окрім OpenCV, існує багато інших технологій розпізнавання образів, таких як TensorFlow, Keras, PyTorch та інші. Ці технології також надають набір інструментів для навчання нейронних мереж та розпізнавання образів, а також можуть бути використані для управління транспортними потоками.

Наприклад, TensorFlow може бути використаний для навчання нейронної мережі для розпізнавання транспортних засобів на зображеннях, а потім використовувати цю модель для розпізнавання транспорту на відео з камер дорожнього руху. Крім того, існують спеціалізовані технології розпізнавання, призначені спеціально для використання в управлінні транспортними потоками, такі як Traficon, яка спеціалізується на розпізнаванні транспортних засобів та інших рухомих об'єктів на дорогах.

1.4 Регресійний аналіз

Регресійний аналіз - розділ математичної статистики, присвячений методам аналізу залежності однієї величини від іншої. На відміну від кореляційного аналізу не з'ясовує істотний зв'язок, а займається пошуком моделі цього зв'язку, вираженої у функції регресії.

Регресійний аналіз використовується в тому випадку, якщо відношення між змінними можуть бути виражені кількісно у виді деякої комбінації цих змінних. Отримана комбінація використовується для передбачення значення, що може приймати цільова (залежна) змінна, яка обчислюється на заданому наборі значень вхідних (незалежних) змінних. У найпростішому випадку для цього використовуються стандартні статистичні методи, такі як лінійна регресія. На жаль, більшість реальних моделей не вкладаються в рамки лінійної регресії. [5]

У контексті управління транспортними потоками, регресійні моделі можуть використовуватися для прогнозування рівня трафіку на дорогах на основі різноманітних факторів, таких як погода, дата та час доби, наявність публічного транспорту, свята та інші фактори, що впливають на популярність дороги.

Одним з найпоширеніших методів регресійного аналізу є лінійна регресія, що передбачає лінійну залежність між залежною та незалежними змінними. Лінійна

регресія може бути використана для побудови моделі прогнозування рівня трафіку на основі зазначених факторів. Також можна використовувати не лінійні регресійні моделі, такі як поліноміальна регресія, яка передбачає не лінійну залежність між залежною та незалежними змінними. Наприклад, поліноміальна регресія може бути використана для прогнозування трафіку на основі квадратичних чи кубічних функцій часу.

1.4.1 Кластерний аналіз

Кластерний аналіз - задача розбиття заданої вибірки об'єктів (ситуацій) на підмножини, які називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися. Завдання кластеризації відноситься до статистичної обробки, а також до широкого класу завдань навчання без вчителя.

Кластерний аналіз — це не якийсь один алгоритм, а загальна задача, для розв'язання якої використовуються різні підходи. Зокрема, алгоритми побудови кластерів можуть суттєво відрізнятись у розумінні того, що відносити в один кластер і як їх ефективно шукати. Серед популярних концепцій кластерів є групи з елементами, які утворюються ґрунтуючись на відстані між ними, на щільності ділянок у просторі даних, інтервалах або на конкретних статистичних розподілах. Тому кластеризація може бути сформульована як задача багатокритеріальної оптимізації. Відповідний алгоритм кластеризації та вибору параметрів (включаючи такі параметри, як функція відстані, порогове значення щільності або кількість очікуваних кластерів) залежать від конкретного набору даних та мети використання результатів. Кластерний аналіз як такий є не автоматизованим завданням, а ітераційним процесом виявлення знань або інтерактивної багатокритеріальної оптимізації, який містить спроби та невдачі. [6]

Цей метод може бути корисним для управління транспортними потоками, оскільки він може допомогти зорієнтуватися в обсязі даних і знайти схожі групи даних, які можуть бути корисні при прийнятті рішень.

Кластерний аналіз може бути використаний для управління транспортними потоками в наступні способи:

1) Сегментація дорожньої мережі: Кластерний аналіз може бути використаний для сегментації дорожньої мережі на різні частини в залежності від їх характеристик, таких як густота трафіку, швидкість, густота населення та інші параметри.
2) Прогнозування трафіку: Кластерний аналіз може бути використаний для прогнозування трафіку на основі характеристик дорожньої мережі та інших зовнішніх факторів.
3) Розподіл ресурсів: Кластерний аналіз може бути використаний для розподілу ресурсів, таких як поліцейські автомобілі, щоб краще контролювати рух транспорту в окремих зонах.
4) Планування маршрутів: Кластерний аналіз може бути використаний для планування оптимальних маршрутів, залежно від характеристик дорожньої мережі та поточної ситуації на дорозі.

Кластерний аналіз може бути використаний для управління транспортними потоками шляхом розподілу транспортних засобів на декілька кластерів за різними ознаками. Наприклад, можна провести кластерний аналіз руху транспорту на основі швидкості руху, частоти гальмування та прискорення, довжини та ширини транспортних засобів та інших факторів, що впливають на рух транспорту.

Отримані кластери можуть бути використані для розподілу транспортних засобів на декілька маршрутів або для оптимізації руху транспорту на певній ділянці дороги. Наприклад, якщо на певній ділянці дороги переважають

транспортні засоби з високою швидкістю руху, то можна збільшити максимально допустиму швидкість на цій ділянці. З іншого боку, якщо на ділянці переважають транспортні засоби з низькою швидкістю руху, то можна знизити максимально допустиму швидкість, щоб уникнути заторів. На основі даних про кількість транспортних засобів, їх швидкість руху та інших факторів можна прогнозувати кількість транспорту, який буде на дорозі в майбутньому. Це допоможе управлінцям дорожнього руху приймати рішення щодо розподілу транспорту на дорогах та зменшення заторів.

1.4.2 Алгоритми глибинного навчання

Глибинне навчання - це підмножина машинного навчання, яка, по суті, є нейронною мережею з трьома або більше рівнями. Ці нейронні мережі намагаються змоделювати поведінку людського мозку — хоча й далеко не збігаються з його можливостями — дозволяючи йому «навчатися» на великих обсягах даних. Хоча нейронна мережа з одним шаром все ще може робити приблизні прогнози, додаткові приховані шари можуть допомогти оптимізувати та покращити точність.

Глибинне навчання керує багатьма додатками та службами штучного інтелекту (AI), які покращують автоматизацію, виконуючи аналітичні та фізичні завдання без втручання людини. Технологія глибокого навчання лежить в основі повсякденних продуктів і послуг (таких як цифрові помічники, пульти дистанційного керування телевізором із голосовою підтримкою та виявлення шахрайства з кредитними картками), а також нових технологій (таких як безпілотні автомобілі).[7]

До найпопулярніших алгоритмів глибинного навчання належать:

1) Згорткові нейронні мережі (Convolutional Neural Networks, CNN) - це тип нейронних мереж, які широко використовуються для розв'язання задач

комп'ютерного зору, зокрема для обробки зображень та розпізнавання об'єктів на зображеннях.

2) Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) - це тип нейронних мереж, які підходять для роботи з послідовністю даних, таких як мовні дані, часові ряди або музичні композиції.
3) Deep Belief Networks (DBNs) - це багатошарові нейронні мережі, які використовуються для вирішення завдань розпізнавання образів та обробки даних. DBNs можуть використовуватися для аналізу траєкторій руху транспортних засобів та прогнозування їх майбутнього руху.
4) Generative Adversarial Networks (GANs) - це архітектури нейронних мереж, які використовуються для генерації нових даних на основі навчального набору даних. GANs можуть бути використані для створення симуляційних моделей транспортного потоку, які допоможуть в аналізі різних сценаріїв та плануванні оптимальних стратегій управління транспортними потоками.

В управлінні транспортними потоками глибинне навчання може бути використане для багатьох завдань. Наприклад, згорткові нейронні мережі можуть використовуватись для автоматичного розпізнавання автомобілів на зображеннях камер дорожнього спостереження. Рекурентні нейронні мережі можуть бути використані для прогнозування трафіку в реальному часі на основі даних з датчиків руху.

1.4.3 Моделі прогнозування

Існує декілька моделей прогнозування дорожнього руху, серед яких можна виділити наступні:

1) ARIMA (Autoregressive Integrated Moving Average) - це статистична модель, яка базується на аналізі попередніх значень часового ряду і використовується

для прогнозування майбутніх значень. ARIMA є ефективним інструментом для прогнозування трафіку на дорогах з низькою і середньою щільністю.

2) LSTM (Long Short-Term Memory) - це рекурентна нейронна мережа, яка добре справляється з прогнозуванням складних часових рядів з довготривалими залежностями. LSTM може бути використаний для прогнозування трафіку на дорогах з високою щільністю, де є багато факторів, що впливають на рух транспорту.
3) Prophet - це відносно нова бібліотека прогнозування часових рядів від Facebook. Вона базується на моделі адитивної регресії зі складовими, що враховують щорічні, щотижневі та денні зміни. Prophet може бути використаний для прогнозування трафіку на дорогах з великою кількістю циклічних змін.
4) XGBoost (Extreme Gradient Boosting) - це алгоритм машинного навчання, який використовується для прогнозування різноманітних показників, включаючи трафік на дорогах. XGBoost може бути використаний для прогнозування трафіку на дорогах з різними типами доріг і з різними вимогами до транспортного потоку.

1.4.4 Системи моніторингу та управління

Системи моніторингу та управління транспортними потоками (СМУТП) включають в себе комплекс різноманітних інструментів, призначених для моніторингу, аналізу та управління дорожнім рухом. Вони можуть використовувати різні методи та технології, включаючи машинне навчання, обробку зображень, геоінформаційні системи та інші.

Одним з головних завдань СМУТП є забезпечення безпеки та комфорту руху транспорту на дорогах. Для цього використовуються різні методи, такі як аналіз потоків, прогнозування руху, виявлення аварійних ситуацій та інші. Системи

моніторингу та управління транспортними потоками можуть також включати в себе інструменти для оптимізації дорожньої мережі та управління заторами. Наприклад, вони можуть визначати оптимальні маршрути для транспортних засобів, розраховувати оптимальні часи для переходу світлофорів та інші.

Для налагодження систем моніторингу транспортних потоків необхідно виконати наступні кроки:

1) Вибір обладнання: необхідно визначитися з вибором камер відеоспостереження, датчиків трафіку та іншого обладнання, яке буде встановлено на дорогах для збору інформації про транспортні потоки.
2) Встановлення обладнання: після вибору необхідного обладнання необхідно встановити його на відповідні ділянки доріг.
3) Підключення до мережі: після встановлення обладнання його необхідно підключити до мережі збору даних, яка забезпечує збір, передачу і аналіз даних.
4) Конфігурування системи: необхідно налаштувати систему моніторингу, встановити необхідні параметри і конфігурації, які відповідають вимогам відповідних організацій та спеціалістів з транспортної безпеки.
5) Аналіз даних: після збору даних, необхідно провести їх аналіз та обробку з використанням відповідних аналітичних інструментів та методів.
6) Прийняття рішень: на основі аналізу даних можна приймати рішення щодо управління транспортними потоками, такі як регулювання світлофорів, розстановка дорожніх знаків, перенаправлення транспорту і т.д.
7) Підтримка системи: необхідно забезпечити регулярне обслуговування та підтримку системи моніторингу, а також оновлювати її з використанням новітніх технологій та розробок.

Узагальнюючи, системи моніторингу та управління транспортними потоками є важливим інструментом для забезпечення безпеки та ефективності дорожнього руху.

1.5 Попередження учасників дорожнього руху про аварійні ситуації

Для попередження учасників дорожнього руху про аварійні ситуації можна використовувати різноманітні засоби, такі як дорожні знаки, світлові табло, мережеві системи сповіщення тощо. Одним з найбільш ефективних засобів є встановлення системи автоматичного сповіщення про аварії на дорозі, яка відправлятиме повідомлення на телефони або інші пристрої водіїв, які знаходяться в зоні ризику. Ця система може бути реалізована на основі аналізу даних з камер спостереження, радарів, GPS-трекерів та інших пристроїв, що встановлюються на дорогах.

Для реалізації такої системи необхідно мати високопродуктивне обчислювальне обладнання та програмне забезпечення для обробки та аналізу великого обсягу даних. Також потрібно забезпечити швидкий доступ до мережі Інтернет та розробити ефективний алгоритм сповіщення про аварії з урахуванням різних факторів, таких як тип дороги, час доби, погода та інші. Для забезпечення надійності та стабільності роботи системи також потрібно мати механізми контролю та діагностики, які дозволять вчасно виявляти та усувати можливі несправності та помилки.

Встановлення дорожніх знаків, камер та світлових табло на дорогах України регулюється законодавством, а саме Правилами дорожнього руху України та Інструкцією з облаштування та експлуатації світлових сигналів та світлових табло. Для встановлення дорожніх знаків та світлових табло на дорогах України необхідно мати дозвіл від відповідних органів. Згідно з Інструкцією з облаштування та

експлуатації світлових сигналів та світлових табло, зазначені пристрої повинні бути встановлені на спеціальних опорах або конструкціях, які забезпечують їх надійність та стійкість.

При встановленні дорожніх знаків, камер спостереження та світлових табло на дорогах необхідно дотримуватися вимог Правил дорожнього руху України та Інструкції з облаштування та експлуатації світлових сигналів та світлових табло. Крім того, при встановленні пристроїв необхідно враховувати характеристики дорожнього руху на конкретній ділянці дороги та забезпечити їх належне функціонування.

1.5.1 Оцінка критеріїв завантаження дорожнього руху у місті

Оцінка критеріїв завантаження дорожнього руху у місті є важливим завданням для планування і управління транспортним потоком. Для цього можуть використовуватися різноманітні методи та інструменти, такі як збір та аналіз даних про рух транспорту, створення математичних моделей для прогнозування навантаження на дороги та інше.

Один з основних критеріїв завантаження дорожнього руху - це потік автотранспорту. Для його оцінки можуть використовуватися спеціальні лічильники, які розміщуються на дорогах та реєструють кількість автомобілів, що проїжджають через певний відрізок дороги за певний проміжок часу. Отримані дані можуть бути використані для оцінки кількості автомобілів на дорозі за годину, день або інший період часу.

Іншим важливим критерієм є швидкість руху транспорту. Для оцінки цього критерію можуть використовуватися так звані радари, які здатні вимірювати швидкість руху автомобілів на дорозі. За отриманими даними можуть бути побудовані математичні моделі, які дозволяють оцінити середню швидкість руху

автомобілів на дорозі, а також визначити місця, де можуть виникати затори та затримки. Для оцінки завантаження дорожнього руху також можуть бути використані різноманітні індикатори, які показують стан трафіку на дорогах у режимі реального часу.

1.6 Перелік необхідних даних про транспортні потоки

Для того, щоб правильно розподіляти трафік і вчасно приймати різного роду рішення для керування транспортними потоками та зменшення завантаженості на дороги необхідні такі дані:

1) Кількість транспортних засобів, які проїжджають певну ділянку дороги протягом певного часу.
2) Швидкість руху транспортних засобів на певних ділянках дороги протягом дня.
3) Дані про аварійність на певних ділянках доріг.
4) Інформація про ремонтні роботи на дорогах.
5) Інформація про тимчасові зміни в організації дорожнього руху.
6) Інформація про погодні умови та інші природні фактори, які можуть впливати на дорожні умови.
7) Дані про вантажні перевезення та інші специфічні види транспорту, що рухаються по дорогах.
8) Дані про місце розташування ділянок доріг з найбільшою кількістю ДТП для забезпечення безпеки руху.
9) Інформація про плани розвитку міста, зміни в інфраструктурі, будівництво нових ділянок доріг та інші фактори, що можуть впливати на дорожній рух.
10) Дані про склад, кількість та маршрути громадського транспорту.

11) Дані про популярність та питому вагу різних видів транспорту серед мешканців міста.

Ці дані можуть бути зібрані за допомогою різних датчиків, які розташовані по всій дорозі, або за допомогою моніторингу суспільного транспорту та мобільних додатків для збору даних про транспорт.

1.7 Оцінка фінансової складової впровадження системи управління

Оцінка фінансової складової впровадження системи управління транспортними потоками на основі машинного навчання залежить від багатьох факторів:

1) Розмір і склад міста: чим більше місто, тим складніше і дорожче буде впровадження системи управління транспортними потоками на основі машинного навчання.
2) Кількість доріг та перехресть: чим більше доріг і перехресть, тим більше обладнання і програмного забезпечення буде потрібно.
3) Тип транспортних потоків: рух вантажівок може вимагати більшої кількості обладнання та ресурсів, ніж рух легкових автомобілів.
4) Існуюча інфраструктура: вартість впровадження системи залежить від того, чи потрібно буде встановлювати нове обладнання або використовувати існуючу інфраструктуру.
5) Використані технології та обладнання: використання новітніх технологій та обладнання може збільшити вартість проекту.
6) Рівень автоматизації: чим більше автоматизації, тим більше буде вартість проекту.

Для оцінки фінансової складової впровадження системи управління транспортними потоками на основі машинного навчання необхідно врахувати

витрати на розробку програмного забезпечення, закупівлю обладнання, інтеграцію з існуючими системами, налагодження та технічне обслуговування. Також потрібно враховувати витрати на навчання персоналу, який буде працювати з системою, та на розробку алгоритмів та моделей для управління транспортними потоками. Однак використання системи управління транспортними потоками на основі машинного навчання може значно знизити витрати на транспортні послуги, покращити безпеку дорожнього руху та зменшити негативний вплив на довкілля, що у майбутньому перевершить початкові вкладення.

1.8 Висновки до частини 1

Управління транспортними потоками вимагає вирішення багатьох проблем, таких як затори, перевантаження доріг, неефективне розподілення ресурсів та безпека дорожнього руху. Машинне навчання є потужним інструментом для аналізу та прогнозування транспортних потоків. Воно дозволяє моделювати складні взаємодії та залежності між різними факторами, що впливають на рух транспорту. Оцінка критеріїв завантаження дорожнього руху у місті дозволяє визначити рівень навантаження на дороги та вулиці. Це включає аналіз трафіку, швидкості руху, пробок, кількості транспортних засобів тощо. Така оцінка допомагає виробляти стратегії та приймати рішення щодо управління транспортними потоками.

Управління транспортними потоками вимагає комплексного підходу та використання різних методів та технологій. Використання машинного навчання, нейронних мереж, розпізнавання рухомих об'єктів, регресійних моделей, кластерного аналізу та інших алгоритмів допомагає ефективно аналізувати, передбачати та управляти транспортними потоками. Системи моніторингу та управління, попередження учасників дорожнього руху та оцінка критеріїв завантаження дорожнього руху також є важливими компонентами управління

транспортними потоками. При впровадженні системи необхідно ретельно оцінювати переваги і недоліки алгоритмів, що допоможе забезпечити ефективне функціонування та покращити управління транспортними потоками.

2. Вибір моделей та алгоритмів

Основною метою дипломної роботи є необхідність навчити світлофори на перехрестях примати рішення - який і коли сигнал світлофору потрібно вмикати, щоб максимізувати кількість автомобілів, які пройдуть через дане перехрестя і мінімізувати час очікування інших автомобілів, які чекають на проїзд. Одним з найкращих методів для цього є навчання з підкріпленням. Також було прийняте рішення покращити цей алгоритм за допомогою своєчасної зміни початкових винагород та покарань в залежності від станів системи.

Навчання з підкріплення і Q-навчання - це два терміни, які часто використовуються сумісно один з одним. Навчання з підкріпленням - це загальна методологія навчання, в якій агент взаємодіє з навколишнім середовищем і отримує нагороду або покарання, в залежності від того, які дії він здійснює.

Q-навчання є одним з допоміжних методів навчання з підкріпленням. В основі цього методу лежить ідея про те, що агент має функцію оцінки дій, відому як Q-функція. Ця функція оцінює, яку нагороду агент може очікувати, виконуючи певну дію у конкретному стані середовища. Q-навчання використовується для навчання агента знаходити найбільш оптимальні дії, щоб максимізувати очікувану загальну нагороду. Основна ідея полягає в тому, щоб оновлювати Q-функцію на основі нагород, які отримує агент в результаті своїх дій.

2.1 Основні інструменти для системи управління

Для розробки обраної моделі було прийнято рішення використовувати мову C# та ігровий двигун Unity (для створення системи будівництва дорожніх шляхів та симуляції трафіку), оскільки:

1. Широке застосування: C# є однією з найпопулярніших мов програмування, особливо в контексті розробки додатків для платформи .NET. Вона має широке спільноту розробників та розгалужену екосистему інструментів, бібліотек і фреймворків, що полегшує розробку та підтримку системи управління транспортними потоками.
2. Об'єктно-орієнтована мова: C# підтримує об'єктно-орієнтований підхід до програмування, що сприяє організації коду у логічні сутності, що відповідають реальним об'єктам та процесам управління транспортними потоками. Це полегшує розуміння, розширення та підтримку системи.
3. Інтеграція з платформою .NET: C# є основною мовою програмування для платформи .NET, яка надає широкі можливості для розробки різноманітних додатків. Це включає в себе підтримку роботи з базами даних, мережевими операціями та іншими функціональними можливостями, які можуть бути корисними при реалізації системи управління транспортними потоками.
4. Безпека та надійність: C# володіє рядом вбудованих механізмів безпеки та контролю типів, що сприяє створенню безпечного та надійного коду. Це особливо важливо при розробці системи, яка взаємодіє з реальними транспортними потоками, де надійність та точність є ключовими факторами.
5. Підтримка машинного навчання: C# має ряд бібліотек та фреймворків, які підтримують машинне навчання, такі як ML.NET та Accord.NET. Це дозволяє легко інтегрувати алгоритми машинного навчання для аналізу даних про транспортні потоки та прийняття рішень щодо управління.

6. Інтеграція з платформою Unity: C# є однією з основних мов програмування для розробки ігор на платформі Unity, що надає багато можливостей для створення віртуального міста, реалістичних симуляцій дорожнього руху та впровадження алгоритмів навчання в цю систему.

2.2 Основні алгоритми для розробленої моделі

Структура Q-таблиці (основні параметри, що необхідні для керування системою): індекс ітерації, кумулятивна винагорода, час простою шляху світлофора, сума часу очікування автомобілів, кількість авто, що проїхали перехрестям, середній час очікування на світлофорі, час зеленого сигналу світлофора для кожної фази.

```
public class QTableData
{
    public int intersectionIteration;
    public float cumulative_reward;
    public float emptyWaitingTime;
    public float sumWaitingTimes;
    public int carsPassed;
    public float avarageCarWaitingTime;
    public List<float> phasesTimers = new List<float>();

    public QTableData(int _phaseIteration, float _cumulative_reward)
    {
        intersectionIteration = _phaseIteration;
        cumulative_reward = _cumulative_reward;
    }

    public QTableData(int _phaseIteration, float _cumulative_reward, float _emptyWaitingTime, float _sumWaitingTimes, int
_carsPassed, float _avarageCarWaitingTime, List<float> _phasesTimers)
    {
        intersectionIteration = _phaseIteration;
        cumulative_reward = _cumulative_reward;
        emptyWaitingTime = _emptyWaitingTime;
        sumWaitingTimes = _sumWaitingTimes;
        carsPassed = _carsPassed;
        avarageCarWaitingTime = _avarageCarWaitingTime;
        phasesTimers = _phasesTimers;
    }
}
```

Калькуляція винагороди (основний метод, який приймає в себе дані про середовище і калькулює нову винагороду): нова винагорода вираховується на основі базової винагороди, кумулятивної винагороди та сталого скаляру. Покарання для часу очікування та часу простою вираховується на основі базового покарання,

кумулятивної винагороди та сталого скаляру. Ці дані додаються та передаються до методу оновлення значень Q-таблиці.

```
private void CalculateReward(QTableData calculatedReward, float jTime, int carsPassed, float emptyWTime, float avgCarWTime,
List<float> phasesTimers)
{
	float reward = 0f;
	float rewardForCarPassed = Mathf.Abs(baseCarPassedReward * calculatedReward.cumulative_reward * rewardScaler);
	float penaltyForWaitingSecond = Mathf.Abs(baseCarWaitingPenalty * calculatedReward.cumulative_reward * rewardScaler);
	float penaltyForEmptyWaitingSecond = Mathf.Abs(baseCarEmptyWaitingPenalty * calculatedReward.cumulative_reward *
rewardScaler);
	reward += rewardForCarPassed * carsPassed;
	reward -= jTime * penaltyForWaitingSecond;
	reward -= emptyWTime * penaltyForEmptyWaitingSecond;
	UpdateQValue(reward, jTime, carsPassed, emptyWTime, avgCarWTime, phasesTimers);
}
```

Оновлення Q-таблиці (метод для оновлення даних у Q-таблиці, в залежності від стану середовища): підрахунок нової кумулятивної винагороди базується на отриманій сумарній винагороді, найвищій або випадковій існуючій кумулятивній винагороді та сталих значеннях дисконтування та фактору навчання.

```
private void UpdateQValue(float reward, float junctionTime, int carsPassed, float emptyWaitingTime, float avarageCarWaitingTime,
List<float> phasesTimers)
{
	int currentDataCount = qTableDatas.Count;
	qTableDatas.Add(new QTableData(IntersectionIteration, reward, emptyWaitingTime, junctionTime, carsPassed,
avarageCarWaitingTime, phasesTimers));
	float currentQ = reward;
	float maxNextQ = qTableDatas.Max(x => x.cumulative_reward);
	float newQ = currentQ + learningRate * (reward + discountFactor * maxNextQ - currentQ);
	qTableDatas[currentDataCount].cumulative_reward = newQ;
}
```

Отримання минулої винагороди (метод дослідження і експлуатації для обрання кращої або випадкової винагороди): алгоритм використовує так звану жадібну стратегію дослідження і використання для пошуку найкращого або випадкового значення кумулятивної винагороди.

```
private QTableData GetQReward()
{
        if (qTableDatas == null)
        {
            qTableDatas = new List<QTableData>();
        }

        if (qTableDatas.Count > 0)
        {
            if (UnityEngine.Random.Range(0.0f, 1.0f) < 0.2) // Exploration
            {
                return qTableDatas[UnityEngine.Random.Range(0, qTableDatas.Count)];
            }
            else // Exploitation - epsilon-greedy strategy! Choose the action with the highest Q-value or random to find out other
occurances
            {
                float maxQ = qTableDatas.Max(x => x.cumulative_reward);
                return qTableDatas.FirstOrDefault(data => Mathf.Abs(data.cumulative_reward - maxQ) < double.Epsilon);
            }
        }
        else
        {
            return new QTableData(-1, 1f);
        }
}
```

Отримання середнього часу очікування авто для шляхів поточної фази: середній час очікування вираховується з суми середніх часів очікувань для кожного шляху поточної фази світлофора поділений на кількість шляхів.

```
    public float GetAvarageWaitingTime(List<StreetJoint> joints)
    {
        float avarageWaitingTime = 0;

        foreach (Vector2Int v in streetsPaths)
        {
            var time = joints[v.x].street.paths[v.y].AvarageWaitingTimeForAllCarsBeforeIntersection;
            avarageWaitingTime += time;
        }

        lastAvarageWaitingTime = avarageWaitingTime / streetsPaths.Count;
        return lastAvarageWaitingTime;
    }
```

Отримання сумарного часу очікування авто для шляхів поточної фази: сумарний час очікування вираховується з суми часів очікувань для кожного шляху поточної фази світлофора.

```
public float GetPathsJunctionTime(List<StreetJoint> joints)
{
    float pathsJunctionTime = 0;

    foreach (Vector2Int v in streetsPaths)
    {
        var time = joints[v.x].street.paths[v.y].SumaryWaitingTimeForAllCarsBeforeIntersection;
        pathsJunctionTime += time;
    }

    lastJunctionTime = pathsJunctionTime;
    return lastJunctionTime;
}
```

Зміна фази світлофора (метод зміни фази в залежності від отриманої винагороди): якщо алгоритм машинного навчання в дії, то наступна фаза обирається за принципом найбільшої кількості автомобілів, які очікують на проїзд перехрестям.

```
private void GoToNextPhase(float junctionTime, int carsPassed, float emptyWaitingTime, float avarageCarWaitingTime, List<float>
phasesTimers)
    {
        if (timersCalc)
        {
            List<float> CarsAmountList = new List<float>();

            for (int a = 0; a < phases.Length; a++)
            {
                float carsAmount = phases[a].GetPathsJunctionTime(joints);
                CarsAmountList.Add(carsAmount);
            }

            float maxValue = 0;
            for (int a = 0; a < CarsAmountList.Count; a++)
            {
                if (CarsAmountList[a] > maxValue)
                {
                    maxValue = CarsAmountList[a];
                }
            }
            var maxIndex = CarsAmountList.IndexOf(maxValue);

            currentPhaseIndex = maxIndex;

            var reward = GetQReward();

            CalculateReward(reward, junctionTime, carsPassed, emptyWaitingTime, avarageCarWaitingTime, phasesTimers);
            RecalculateAllTimers(reward);
            IntersectionIteration++;
        }
        else
        {
            if (currentPhaseIndex < phases.Length - 1)
            {
                currentPhaseIndex++;
            }
            else
            {
                currentPhaseIndex = 0;

                var reward = GetQReward();

                CalculateReward(reward, junctionTime, carsPassed, emptyWaitingTime, avarageCarWaitingTime, phasesTimers);
                RecalculateAllTimers(reward);
                IntersectionIteration++;
            }
        }
    }
```

Отримання сумарного часу простою для шляхів поточної фази: сумарний час простою вираховується з суми часів простою для кожного шляху поточної фази світлофора.

```
private void CalculateEmptyWaitingTime()
{
   if (phases.Length != 0)
   {
      foreach (Vector2Int v in phases[currentPhaseIndex].streetsPaths)
      {
         var time = joints[v.x].street.paths[v.y].SumaryWaitingTimeForAllCarsBeforeIntersection;
         var carsInQueue = joints[v.x].street.paths[v.y].enterQueue;
         if (time <= 0f || carsInQueue == 0)
         {
            lastEmptyWaitingTime += Time.deltaTime;
         }
      }
   }
}
```

Перерахунок часу фаз світлофора: перерахунок таймерів для кожної фази відбувається після зміни кожної фази, нове значення вираховується з минулого значення таймеру, кумулятивної винагороди та скаляру, який є дільником загального часу на поточний.

```
private void RecalculateAllTimers(QTableData qTableData)
{
        if (timersCalc)
        {
           RecalculateFullCycleTime();
           float allTime = 0;
           float oneTen = 0.1f * FullCycleTime;

           foreach (IntersectionPhase p in phases)
           {
              var junctionTime = p.GetPathsJunctionTime(joints);
              allTime += junctionTime < oneTen ? oneTen : junctionTime;
           }

           var timers = qTableData.phasesTimers;
           for (int i = 0; i < phases.Length; i++)
           {
              var junctionTime = phases[i].GetPathsJunctionTime(joints);
              var scale = junctionTime / timers[i];
              float newGreenLightTime = (timers[i] - qTableData.cumulative_reward / scale);
              phases[i].phaseGreenLightTime = (float)Math.Round(newGreenLightTime, 3);
           }
        }
}
```

Структура вихідних даних (структура результативних даних для подальшого аналізу роботи системи): структура вихідних даних містить у собі всю необхідну інформацію для проведення аналізу роботи алгоритму, таку як середня дистанція шляху, кількість авто, середня кількість авто, результати Q-таблиці для кожної ітерації перехрестя та інше.

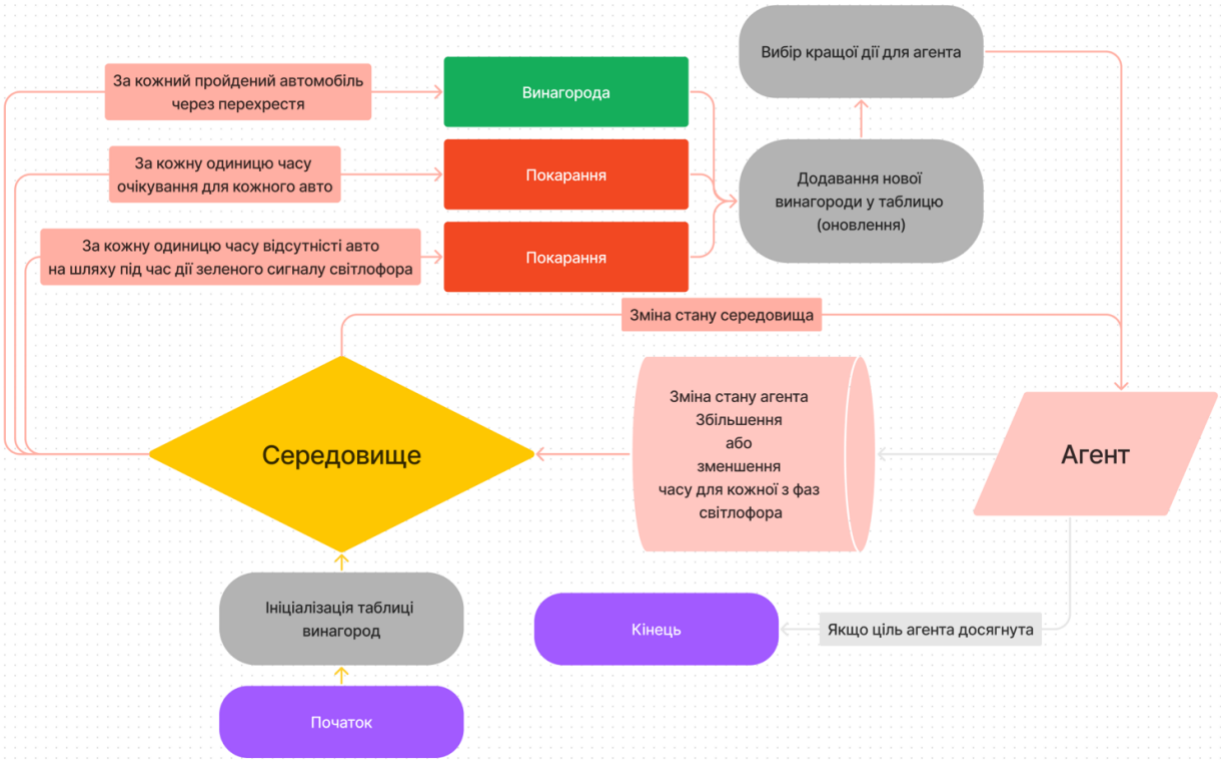
```
public class IterationResultData
    {
        public SimulationType simulationType;
        public int intersectionAmount;
        public float simulationTime;
        public float avarageCarPathDistance;
        public float avarageCarPathCompleteTime;
        public int overallCarsAmount;
        public int avarageCarsAmountAtATime;
        public List<IntersectionResultData> intersectionResultDatas = new List<IntersectionResultData>();

        public IterationResultData(SimulationType _simulationType, int _intersectionAmount, float _simulationTime, float
_avarageCarPathDistance, float _avarageCarPathCompleteTime, int _overallCarsAmount, int _avarageCarsAmountAtATime,
List<IntersectionResultData> _intersectionResultDatas)
        {
            simulationType = _simulationType;
            intersectionAmount = _intersectionAmount;
            simulationTime = _simulationTime;

            avarageCarPathDistance = _avarageCarPathDistance;
            avarageCarPathCompleteTime = _avarageCarPathCompleteTime;

            overallCarsAmount = _overallCarsAmount;
            avarageCarsAmountAtATime = _avarageCarsAmountAtATime;

            intersectionResultDatas = _intersectionResultDatas;
        }
    }

    public class IntersectionResultData
    {
        public int intersectionIndex;
        public List<QTableData> qTableDatas = new List<QTableData>();

        public IntersectionResultData(int _intersectionIndex, List<QTableData> _qTableDatas)
        {
            intersectionIndex = _intersectionIndex;
            qTableDatas = _qTableDatas;
        }
    }
```

2.2.2 Навчання створеної моделі

При навчанні з підкріпленням для управління транспортними потоками ітерації (або епохи) є частинами процесу навчання, які використовуються для покращення дій агента (перехрестя), що взаємодіє з транспортною системою. (Рисунок 1.1)

Кожна ітерація складається з наступних кроків:

1. Стан: агент отримує інформацію про поточний стан транспортної системи, таку як: кількість автомобілів, що чекають на проїзд перехрестям, швидкість руху транспортних засобів, час очікування тощо.
2. Дія: на основі інформації про поточний стан, агент здійснює дію, а саме встановлює відповідні сигнали світлофорів для кожної з фаз на певний час.
3. Нагорода: після того, як агент здійснив дію, він отримує нагороду або покарання, залежно від того, скільки автомобілів проїхали через перехрестя, скільки автомобілів продовжують чекати та скільки часу фаза світлофора простоює.
4. Оновлення Q-таблиці – використовуючи дослідження та експлуатацію (жадібну стратегію) обираємо дію з найвищим значенням Q або випадкову, щоб алгоритм мав шанс прорахувати інші варіанти.
5. Оновлення: на основі отриманої нагороди, агент оновлює свої знання про транспортну систему і виробляє нову стратегію дій для наступної ітерації – збільшує або зменшує час для кожної фази світлофорів.
6. Повторення: процес повторюється для кожної ітерації, з метою навчити агента здійснювати більш оптимальні дії в керуванні транспортною системою.



(Рисунок 1.1)

Кількість ітерацій залежить від розміру транспортної системи, складності задачі, а також від ефективності навчання. Навчання може продовжуватися до тих пір, поки агент здатний здійснювати оптимальні дії в управлінні транспортною системою.

Початкові дані для навчання, винагороди та покарання:

1. Кількість одиниць часу на навчання – 36000
2. Кількість одиниць часу між появою авто – 0.5
3. Максимальна кількість одночасних авто у всьому середовищі – 200-400
4. Винагорода за авто, яке проїхало перехрестя – 1.5
5. Покарання за кожну одиницю часу очікування для кожного авто – 0.0002
6. Покарання за відсутність авто на шляху під час дії зеленого сигналу світлофора – 0.001

7. Швидкість навчання – 0.3 (зазвичай використовується в алгоритмах навчання з підкріпленням, таких як Q-навчання, щоб контролювати ступінь, до якого нова інформація перекриває наявні знання. Він визначає баланс між новою інформацією про винагороду та поточними значеннями Q)

8. Коефіцієнт дисконтування – 0.9 (використовується для визначення важливості майбутніх винагород порівняно з негайними винагородами. Він впливає на процес прийняття рішення агентом, призначаючи вагу майбутнім винагородам. 0 - означає, що агент дбає лише про негайні винагороди, 1 означає, що агент вважає як негайні, так і майбутні винагороди однаково важливими. Значення від 0 до 1 дозволяють знайти компроміс між негайними та майбутніми винагородами. Вищий дисконтний коефіцієнт надає більшої ваги майбутнім винагородам, заохочуючи агента приймати рішення, які максимізують сукупні винагороди з часом. З іншого боку, нижчий дисконтний коефіцієнт робить більший акцент на негайні винагороди, що може призвести до більш вузького прийняття рішень.

2.3 Розробка середовища для тестування моделі

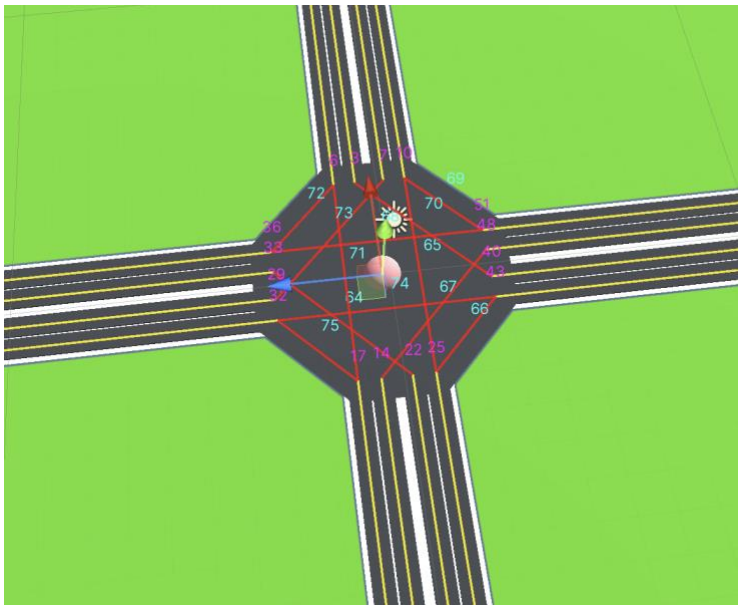
Оскільки встановлення камер спостереження на дорозі не можливе без участі відповідних органів влади, тому було прийнято рішення не розроблювати систему розпізнавання транспортних засобів на зображеннях, а зосередитись саме на регулюванні перехресть. Для того, щоб перевірити якість роботи алгоритму, було розроблено підсистему побудови віртуальної частини міста та підсистему симуляції трафіку.

Алгоритм побудови середовища дозволяє створити майже будь-яку частину міста: дороги з одностороннім рухом, двостороннім рухом, автомагістралі, перехрестя різного типу та інші (Рисунок 1.2). При побудові відрізку дороги

автоматично створюються шляхи для автомобілів, в залежності від обраного типу дороги та кількості смуг.

(Рисунок 1.2)

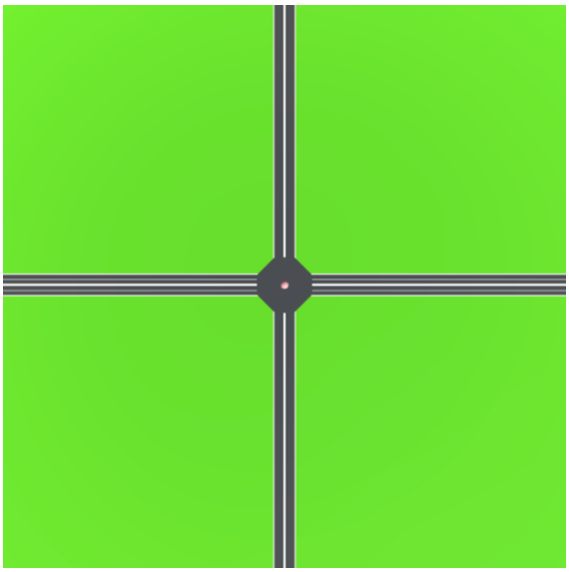
Кожен відрізок дороги має набір нодів, які складаються в шлях, вхідні та вихідні ноди, які поєднуються з перехрестями або є початком чи кінцем руху автомобілів. Роздоріжжя також мають набір вхідних та вихідних нод, які поєднані з вулицями та з яких утворюються шляхи для автомобілів. (Рисунок 1.3) Кожна нода має свій ідентифікатор за яким вона може бути знайдена у тому чи іншому об’єкті.



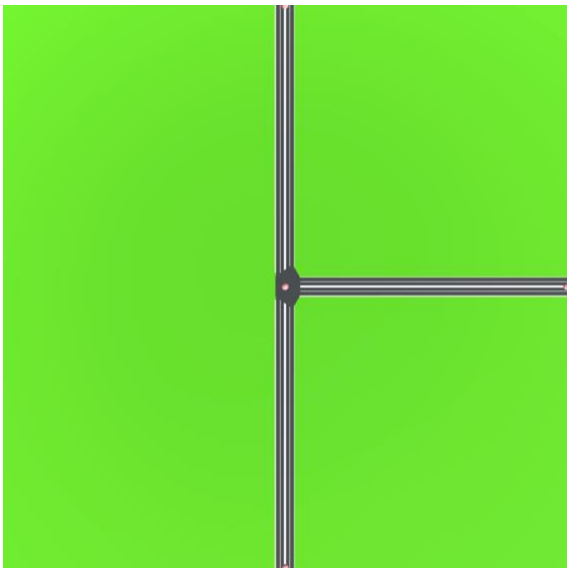
(Рисунок 1.3)

Також кожне перехрестя може мати або не мати світлофори, які розподілені по фазам руху автомобілів, наприклад для Т-перехрестя з двостороннім рухом, по 2 полоси з кожного боку дороги - буде існувати 3 фази світлофорів, а для Х-перехрестя з двостороннім рухом, по 2 полоси з кожного боку дороги – буде існувати 4 фази світлофорів. Для тестування алгоритму було створено 2 варіанти:

Х-перехрестя (4 фази) (Рисунок 1.4) та Т-перехрестя (3 фази) (Рисунок 1.5)



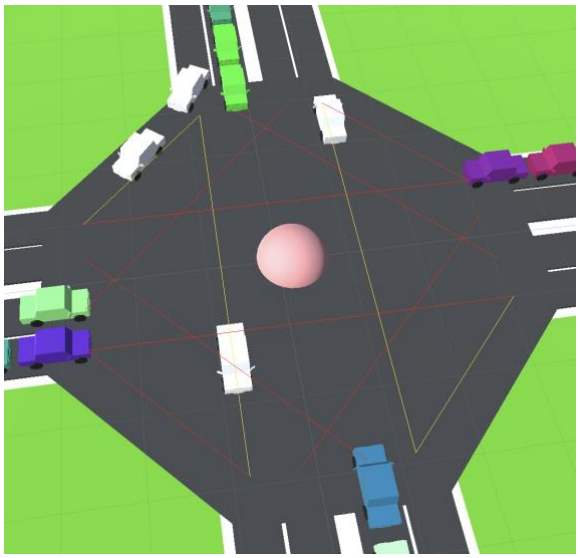
(Рисунок 1.4)



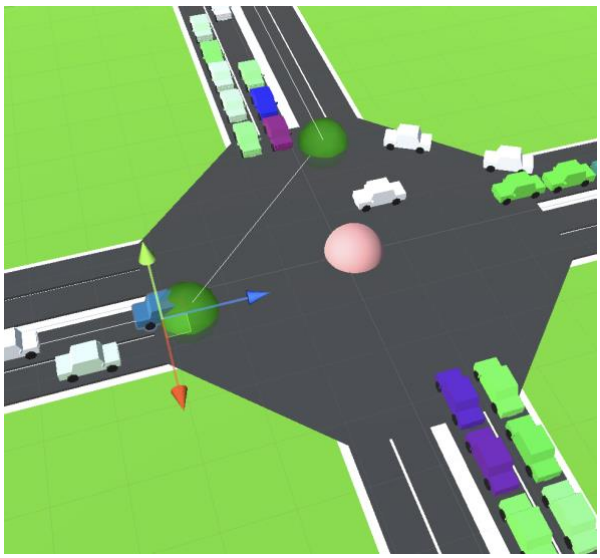
(Рисунок 1.5)

Зміна фаз світлофора відбувається за принципом найбільшого навантаження на шляхи відповідної фази. Тобто, якщо сумарно час очікування на всіх шляхах фази найбільший, то ця фаза і буде обрана у якості наступної. (Рисунок 1.6)

Симуляція трафіку передбачає рівномірно розподілену появу автомобілів у визначених нодах. Автомобілі намагаються обрати найкоротший шлях до своєї кінцевої точки, де початкова та кінцева ноди обираються навмання, проте довжина шляху повинна бути не менша, ніж визначена константним значенням.(Рисунок 1.7)



(Рисунок 1.6)



(Рисунок 1.7)

2.4 Наявні недоліки обраного алгоритму

Навчання з підкріпленням та Q-навчання є потужними інструментами для управління транспортними потоками, але вони також мають свої недоліки:

1. Велика кількість тренування: навчання з підкріпленням та Q-навчання вимагають тренування моделі в реальному часі та на великій кількості ітерацій, що може бути часо-затратним та ресурсо-затратним процесом.
2. Висока складність налаштування великих кількості параметрів, таких як швидкість навчання, функція нагороди, фактор знижки, початкові

винагороди, покарання та інше, що може бути нетривким, потребувати експертних рішень та багато часу на вибір досконалих значень.

3. Велика обчислювальна складність: навчання з підкріпленням та Q-навчання можуть вимагати значних обчислювальних ресурсів, особливо при великих масштабах системи управління транспортними потоками.
4. Непередбачуваність поведінки агентів: управління транспортними потоками включає взаємодію з різними сутностями середовища, такими як автомобілі різного типу, велосипедисти, пішоходи, аварійні ситуації, тому можливість прогнозування поведінки інших агентів може бути обмежена, що може призводити до недоцільних рішень в управлінні транспортними потоками.
5. Обмежена робота в динамічних середовищах: управління транспортними потоками, особливо великими містами, може бути дуже динамічним та швидкозмінним, що може пропонувати великі виклики для моделей, натренованих на статичних даних.

Незважаючи на ці недоліки, навчання з підкріпленням залишаються ефективними підходами для управління транспортними потоками, і вони можуть бути постійно вдосконалені та оптимізовані для кращої пристосованості до реальних умов дорожнього руху.

2.5 Висновки до частини 2

Для досягнення поставленої мети роботи по розробці системи управління транспортними потоками на основі машинного навчання важливо було звернути увагу на вибір моделей та алгоритмів, навчання моделі, розробку середовища для тестування та виявлення, вирішення недоліків. Було створено підсистему побудови віртуального міста та симуляцію трафіку. Була розроблена модель управління транспортними потоками на основі машинного навчання з підкріпленням та Q-

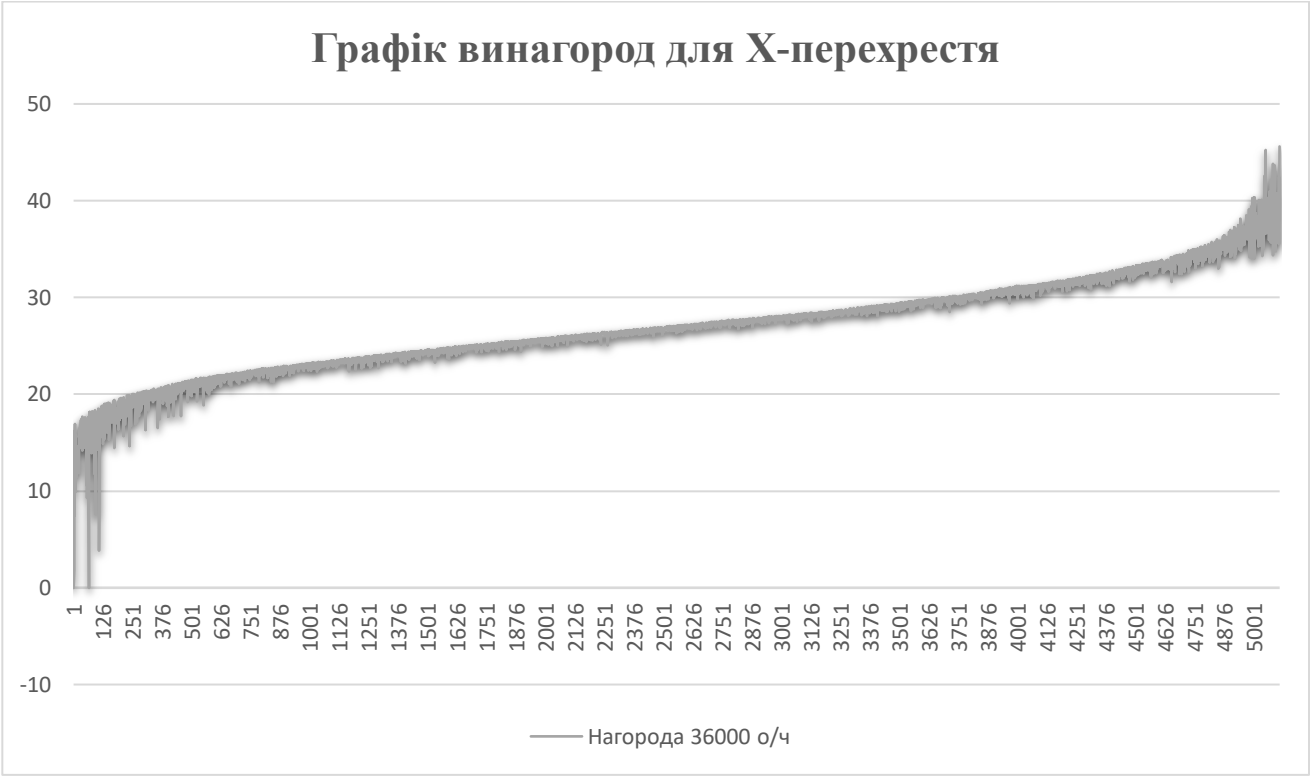
навчанням. Зроблений вибір допоміг забезпечити доволі ефективне управління транспортними потоками та зменшення навантаження на дороги у якості зменшення часу очікування на світлофорі та збільшення кількості автомобілів, які проїжджають перехрестя за одну фазу.

3 Аналіз розробленої моделі та порівняння з ручним керуванням

Проаналізувавши отримані дані, (Таблиця 1 та Таблиця 2) можна зробити висновок, що при однаковому часі виконання симуляції для рівномірно розподіленого трафіку автоматичне керування світлофорами дозволило значно зменшити середній час очікування та середню кількість автомобілів, які чекають проїзду на перехресті. Для Х-перехрестя і Т-перехрестя час очікування зменшився більш ніж у 2 рази, а середня кількість автомобілів, які очікують на проїзд перехрестям зменшилась у 1.6 разів. Також за результатами симуляції з машинним навчанням було знайдено кращі часові фази для отримання вищої винагороди в автоматичному керуванні світлофорами. Можна виділити невеликі коливання винагороди на графіку, це обумовлено тим, що алгоритм час від часу обирає випадкові значення з Q-таблиці. (Графік 1 та Графік 2) Це називається жадібним алгоритмом, який використовує метод вибору випадкового значення з визначеною ймовірністю (дослідження), а в інших випадках найкращу винагороду з минулих ітерацій (експлуатація). Також можна зрозуміти, що коливання винагороди буде збільшуватись або зменшуватись відносно густини трафіку.

Х-перехрестя	Час навчання алгоритму	Середній час очікування на світлофорі	Середня кількість авто, які очікують на світлофорі	Час фаз для вищої винагороди
Ручне керування світлофорами	36000 одиниць часу	26.256483	42	5 о/ч 5 о/ч 5 о/ч 5 о/ч
Автоматичне керування світлофорами	36000 одиниць часу	12.3771086	21	4.113 о/ч 6.488 о/ч 4.047 о/ч 6.136 о/ч

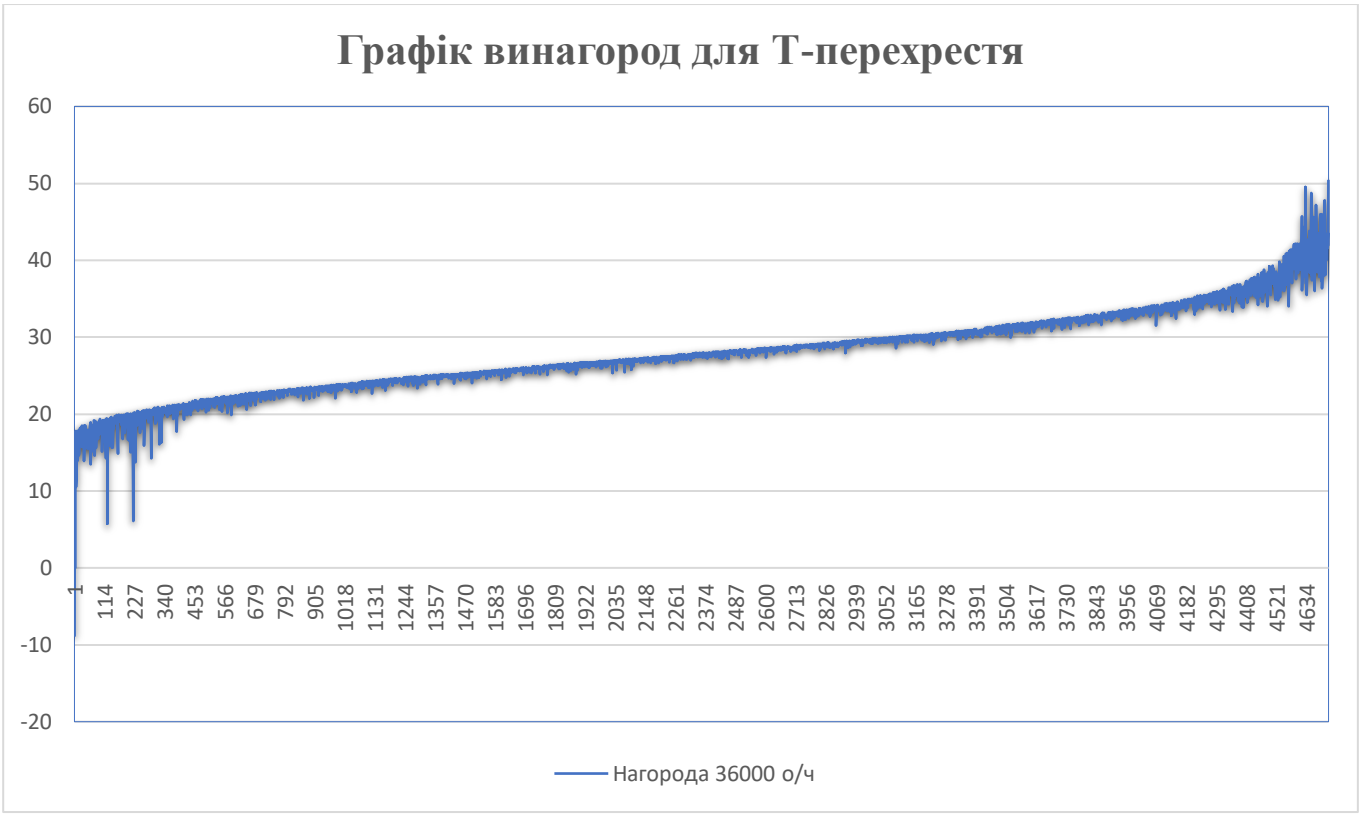
(Таблиця 1)



(Графік 1)

Т-перехрестя	Час навчання алгоритму	Середній час очікування на світлофорі	Середня кількість авто, які очікують на світлофорі	Час фаз для вищої винагороди
Ручне керування світлофорами	36000 одиниць часу	22.482451	31	5 о/ч 5 о/ч 5 о/ч
Автоматичне керування світлофорами	36000 одиниць часу	10.9699736	17	4.73 о/ч 6.077 о/ч 3.751 о/ч

(Таблиця 2)



(Графік 2)

При тестуванні системи було виявлено цікаву закономірність, а саме збільшення сукупного часу дії світлофора призводить до нерівномірної дії алгоритму, це відбувається через те, що занадто довга фаза призводить до скупчення великої кількості автомобілів, що означає кратне збільшення покарання за очікування. При початковому часі дії світлофора > 10 о/ч, алгоритм починає швидко отримувати великі покарання і вже через 20-40 ітерацій повертає час до середніх 5 о/ч на дію фази світлофора. При протилежній ситуації, коли початковий час дії світлофора < 2 о/ч, алгоритм доволі тривалий час отримує дуже малі винагороди і починає збільшувати час дії фази. Такого роду результати вказають на правильність роботи алгоритму та звертає увагу на те, що початкові дані дуже сильно впливають на швидкість розв'язку поставленої задачі. Тому при розробці реальної системи дуже важливо мати сукупність початкових даних, які можна зібрати за допомогою різного роду датчиків встановлених у місті. Також можна помітити, що після 36000 о/ч система починає отримувати винагороди, які дуже сильно різняться між ітераціями, це пов'язано з тим, що при наданих умовах алгоритм вже здобув максимально можливу винагороду. При рівномірному збільшенні кількості автомобілів, так само рівномірно збільшується винагорода. Так само було протестовано алгоритм при різних ситуаціях, наприклад при збільшенні кількості трафіку у 2 рази, середній час дії фази світлофора збільшується у ~1.4 рази. Тобто для Х-перехрестя при ти самих умовах, крім кількості трафіку, фази світлофора при кращій винагороді були: 5.382 о/ч, 8.203 о/ч, 7.124 о/ч, 6.175 о/ч. Звичайно, середній час очікування різниться і становить 17.6423 о/ч при згаданих вище умовах. Можна зробити висновок, що кожне перехрестя має бути навчене окремо на різній кількості трафіку та різних умовах, щоб сумарно отримати найкращий результат.

3.1 Можливі покращення створеної системи управління

Звичайно, ніяка система не є абсолютно досконалою і завжди є можливості для поліпшень. Незалежно від того, як добре розроблена або ефективна система, завжди можна знайти аспекти, які можна покращити або оптимізувати. Це відбувається з кількох причин. По-перше, системи зазвичай розробляються з урахуванням обмежень, які можуть бути наявні на час розробки. Ці обмеження можуть стосуватися ресурсів, часу, бюджету або інших факторів. По-друге, іноді системи виявляються недосконалими через змінні умови або нові вимоги. Швидкі технологічні зміни, зміна потреб користувачів або поява нових викликів можуть змусити переглянути існуючі системи і знайти способи їх поліпшити. Це означає, що з часом можуть з'являтися нові підходи та кращі методи, які можна застосувати до існуючих систем, щоб покращити їх ефективність, безпеку або зручність використання.

У даному випадку можна розглянути використання більш складних алгоритмів навчання з підкріпленням або глибинного навчання для досягнення ще кращих результатів управління транспортними потоками. Вдосконалення алгоритмів може включати оптимізацію різноманітних параметрів, використання нейронних мереж або більш складних моделей. Також можна додати інші дані, такі як: інформація про погодні умови, дорожні події, громадський транспорт тощо. Ці додаткові дані можуть допомогти більш точно передбачати та адаптуватися до змін у дорожніх умовах. У реальних умовах необхідно буде використовувати камери, сенсори та системи збору даних, які будуть встановлені для отримання більш повної та точної інформації про стан дорожнього руху. Це може допомогти виявляти проблемні ділянки, відстежувати рух транспорту та забезпечити більш ефективне управління потоками. Ще одним важливим покращенням можна вважати

поєднання даних з різних світлофорів, завдяки цьому алгоритм зможе більш точно передбачати траєкторії руху транспорту, та завчасно виставляти необхідні таймери. Більш того треба пам’ятати, що автомобілі не єдині учасники дорожнього руху, пішоходи, велосипедисти та інше також дуже сильно впливають на систему управління в цілому.

Ці покращення можуть сприяти більш ефективному управлінню транспортними потоками, зменшенню заторів, скороченню часу очікування та поліпшенню загальної дорожньої ситуації. Враховуючи результати симуляції та аналіз винагород, можна спрямувати зусилля на вдосконалення конкретних аспектів системи управління, щоб досягти ще кращих результатів.

Висновки

Використання машинного навчання з підкріпленням та Q-навчання є потужним підходом до розв'язання задач управління транспортними потоками. Воно дозволяє створити адаптивні алгоритми, які можуть самостійно вчитися і покращувати свої результати з часом. Застосування цих методів дозволяє досягти оптимального розподілу ресурсів, зменшити час очікування на світлофорах, знизити загальну завантаженість доріг та покращити рух транспорту. Важливо підібрати правильні параметри та налаштувати систему навчання з підкріпленням для конкретної задачі управління транспортними потоками. Це може вимагати додаткового експериментування та оптимізації. Потрібно мати якісні дані про дорожні умови, рух транспорту, вимоги та інші фактори, щоб ефективно навчити алгоритми з підкріпленням. Це включає збір інформації з датчиків, камер спостереження, систем моніторингу тощо. Для ефективного використання навчання з підкріпленням та Q-навчання необхідно використовувати потужні обчислювальні ресурси та оптимізовані алгоритми, оскільки обробка великих обсягів даних може бути вимогливою з точки зору обчислювальної потужності. Результати дослідження та впровадження системи управління транспортними потоками на основі навчання з підкріпленням та Q-навчання можуть бути об'єктом подальших досліджень та покращень. Варто розглядати можливості додаткової оптимізації та розширення функціональності системи. Успішне впровадження системи управління транспортними потоками може принести значні переваги, такі як покращення ефективності дорожнього руху, зниження витрат палива, зменшення заторів, зменшення забрудненості міста та покращення загальної мобільності.

Список використаних джерел

1. https://uk.wikipedia.org/wiki/Машинне_навчання
2. https://uk.wikipedia.org/wiki/Навчання_з_підкріпленням
3. https://www.ibm.com/topics/neural-networks
4. https://opencv.org/
5. https://uk.wikipedia.org/wiki/Регресійний_аналіз
6. https://uk.wikipedia.org/wiki/Кластерний_аналіз
7. https://www.ibm.com/topics/deep-learning