ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка методу автоматизованого керування автотранспортом за допомогою штучного інтелекту на прикладі відеогри»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
(код, найменування спеціальності)
освітньо-професійної програми «Інженерія програмного забезпечення»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Валерій ГАВРИЛЮК
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-64
Валерій ГАВРИЛЮК

Керівник: Наталія ТРІНТІНА
к.т.н., доцент

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гаврилюку Валерію Олександровичу

1. Тема кваліфікаційної роботи: «Розробка методу автоматизованого керування автотранспортом за допомогою штучного інтелекту на прикладі відеогри»

керівник кваліфікаційної роботи Наталія ТРІНТІНА к.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» жовтня 2023 р. №145.

2. Строк подання кваліфікаційної роботи «29» грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, методи машинного навчання, алгоритм Q-навчання, стани системи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд предметної області
2. Розробка моделей та методів
3. Розробка програмного забезпечення моделей

4. Проведення моделювання та аналіз отриманих результатів

5. Перелік графічного матеріалу: *презентація*
   1. Мета, об`єкт та предмет дослідження
   2. Порівняння парадигм машинного навчання
   3. Математична модель орієнтування автотранспорту в середовищі
   4. Схема методу автоматизованого керування автотранспорту за допомогою штучного інтелекту
   5. Результат порівняння методів
   6. Практичний результат
   7. Висновки

6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз науково-технічної літератури	19.10-05.11.23	
2	Вивчення матеріалів для аналізу розвитку технологій машинного навчання	06.11-12.11.23	
3	Вибір алгоритму для навчання моделі автомобіля	13.11-19.11.23	
4	Розробка середовища для автомобіля	20.11-26.11.23	
5	Програмування логіки поведінки автомобіля	27.11-03.12.23	
6	Тестування методу та проведення досліджень	04.12-10.12.23	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.23	
8	Розробка демонстраційних матеріалів	21.12-29.12.23	

Здобувач вищої освіти _____ Валерій ГАВРИЛЮК
(підпис)

Керівник
кваліфікаційної роботи _____ Наталія ТРІНТІНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 72 стор., 1 табл., 26 рис., 31 джерело.

Мета роботи – мінімізація відхилення від заданої траекторії в процесі автоматизованого керування автотранспортом.

Об'єкт дослідження – процес автоматизованого керування автотранспортом.

Предмет дослідження – Методи автоматизованого керування автотранспортом за допомогою штучного інтелекту.

Короткий зміст роботи: У роботі було проаналізовано існуючі підходи до побудови та використання штучного інтелекту у сфері моделювання руху автотранспорту. Після аналізу, було обрано за основу розроблюваного методу алгоритм Q-навчання. Було розроблено метод автоматизованого керування автотранспорту та середовище для тренування розробленої моделі. Проведено аналіз отриманих результатів моделювання та визначено переваги використання розробленого методу, для досягнення мети мінімізації відхилення від заданої траекторії під час автоматизованого керування автотранспортом.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ КЕРУВАННЯ АВТОМОБІЛЕМ, Q-НАВЧАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, СТАНИ, РОБОЧЕ СЕРЕДОВИЩЕ, ОПТИМІЗАЦІЯ ДІЙ АГЕНТА.

ABSTRACT

Text part of the master's qualification work: 72 pages, 26 pictures, 1 table, 31 sources.

The purpose of the work is to minimize the deviation from the given trajectory in the process of automated vehicle control.

The object of research is the process of automated vehicle control.

Subject of research – is methods of automated vehicle control using artificial intelligence.

Summary of the work: The work analyzed existing approaches to the construction and use of artificial intelligence in the field of vehicle traffic simulation. After the analysis, the Q-learning algorithm was chosen as the basis of the developed method. A method of automated vehicle control and an environment for training the developed model were developed. The obtained simulation results were analyzed and the advantages of using the developed method were determined in order to achieve the goal of minimizing the deviation from the given trajectory during automated vehicle control.

KEYWORDS: CAR DRIVING AUTOMATION, Q-LEARNING, ARTIFICIAL INTELLIGENCE, STATES, WORKING ENVIRONMENT, AGENT ACTION OPTIMIZATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 9
ВСТУП 10
РОЗДІЛ 1 АНАЛІЗ НАУКОВО-ТЕХНІЧНОЇ ЛІТЕРАТУРИ 12
1.1. Машинне навчання 12
1.2. Аналіз технологій автономних систем транспортних засобів 15
РОЗДІЛ 2 АНАЛІЗ ТА ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ 24
2.1 Навчання з підкріпленням та Deep Q Network 24
2.1.1 Навчання з підкріпленням 24
2.1.2 Deep Q Network 29
2.2 Огляд аналогів 36
2.2.1 Навчання з учителем 36
2.2.2 Навчання без учителя 41
РОЗДІЛ 3 РЕАЛІЗАЦІЯ МЕТОДУ АВТОМАТИЗОВАНОГО КЕРУВАННЯ АВТОТРАНСПОРТОМ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ 44
3.1 Аналіз сучасних засобів програмної інженерії для реалізації методу автоматизованого керування автотранспортом 44
3.1.1 TensorFlow 44
3.1.2 NumPy 48
3.2 Реалізація методу автоматизованого керування автотранспорту 58
3.3 Аналіз результатів 68
ВИСНОВКИ 72
ПЕРЕЛІК ПОСИЛАНЬ 73
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація) 76

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ML	-	Machine Learning
DQN	-	Deep Q Network
RL	-	Reinforcement Learning
SL	-	Supervised Learning
UL	-	Unsupervised learning

ВСТУП

У наш час автомобільна індустрія неперервно розвивається і одним із видів розвитку є створення систем автоматизованого керування автотранспортом. Підвищення ефективності автоматизованого водіння, включаючи застосування новітніх технологій штучного інтелекту, робить можливим швидке поширення безпілотних машин.

Автоматизовані автомобілі - це транспортні засоби з певним рівнем самостійного управління, досягнутого за допомогою складних систем програмного забезпечення та апаратної бази, які забезпечують безпечне керування усіма або окремими аспектами автомобільного руху.

Основною перевагою, окрім автоматизованого водіння, виступають додаткові функції безпеки, такі як екстрене гальмування, автопілот та інші. Коли штучний інтелект впливає на прийняття рішень водієм, ми маємо справу з автоматизованим транспортом. Для цього машини використовують датчики та камери для отримання інформації про навколишнє середовище. Залучений штучний інтелект обробляє ці дані для прийняття рішень. Датчики реагують на рух інших автомобілів, а камери розпізнають світлофори, пішоходів і інші об'єкти дорожнього руху. Технологія лідару, що використовується для вимірювання відстаней та виявлення світла, сприяє ще більшій точності в передачі інформації. Алгоритми автомобіля фільтрують інформацію для розпізнавання об'єктів та прогнозування руху. Отримані дані надходять до системи штучного інтелекту автомобіля, який контролює рух, гальмування та режим прискорення.

Зважаючи на стрімкий технологічний розвиток у сфері автомобільної промисловості, впровадження автономних систем керування автотранспортом стає ключовим етапом еволюції цієї галузі. Розробка та застосування методів штучного інтелекту у керуванні транспортними засобами відкриває нові можливості в покращенні безпеки дорожнього руху та ефективності транспортних систем. Актуальність таких досліджень визначається потребою у створенні надійних та безпечних систем автоматизованого керування, що відповідають

вимогам сучасного технологічного прогресу та розвитку мобільності.

Мета роботи – мінімізація відхилення від заданої траєкторії в процесі автоматизованого керування автотранспортом.

Об'єкт дослідження – процес автоматизованого керування автотранспортом.

Предмет дослідження – Методи автоматизованого керування автотранспортом за допомогою штучного інтелекту.

1 АНАЛІЗ НАУКОВО-ТЕХНІЧНОЇ ЛІТЕРАТУРИ

1.1. Машинне навчання

Машинне навчання - це галузь штучного інтелекту, яка дозволяє комп'ютерам навчатися і покращувати свою продуктивність без явного програмування. Вона базується на розвитку алгоритмів та моделей, які дають змогу системам аналізувати дані, розпізнавати патерни в них і здійснювати прогнози або приймати рішення на їх основі.[1]

У машинному навчанні використовуються різні типи парадигм, серед яких найвідоміші - навчання з учителем, ненавчане навчання та підсилене навчання.

Навчання з учителем (supervised learning) - це один із основних видів машинного навчання, де модель навчається на основі розмічених даних. В цьому методі модель вивчається залежностей і закономірностей між вхідними даними (ознаками) та відповідними мітками чи цільовими значеннями.[2]

Процес навчання з учителем полягає у передачі моделі великої кількості прикладів, де для кожного вхідного набору даних (ознак) є відоме відповідне правильне значення (мітка). (рисунок 1.1.)

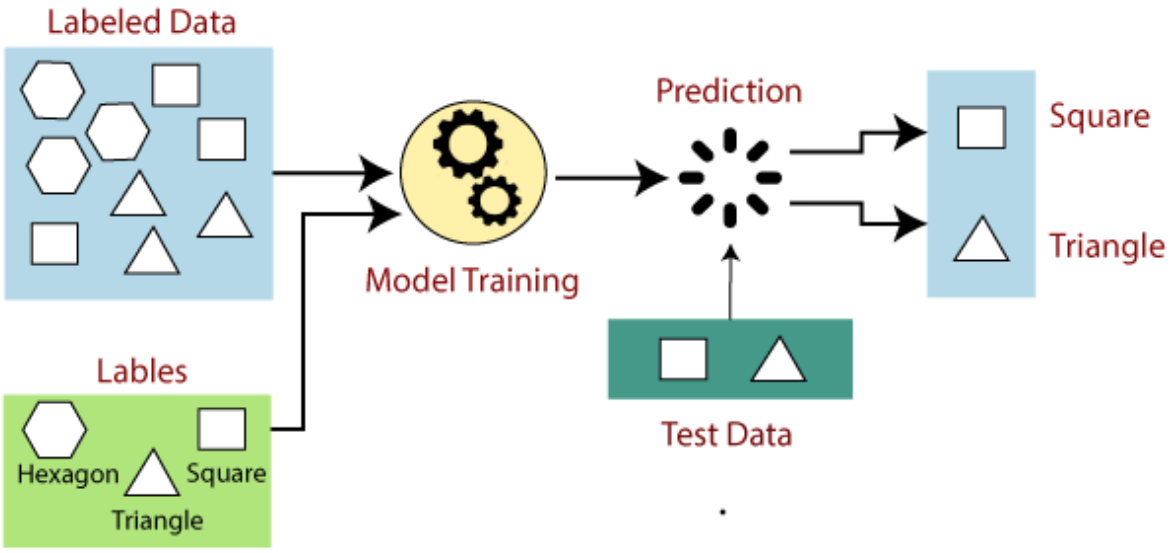


Рис 1.1. Схема роботи методу навчання з учителем.

Модель навчається встановлювати зв'язки між вхідними даними та їх відповідними вихідними значеннями, щоб у майбутньому здати точні прогнози для нових, раніше невідомих даних.

Цей тип навчання використовується для завдань прогнозування, класифікації, регресії та інших. Наприклад, у випадку класифікації, модель навчається розпізнавати категорії або класи на основі вхідних даних та відповідних міток.[2] У завданнях регресії модель прогнозує числові значення на основі залежностей між ознаками та цільовими значеннями.

Метод навчання з учителем використовується в різних сферах, таких як медицина, фінанси, маркетинг, комп'ютерний бачення та інші, де важливо мати точні та достовірні прогнози або класифікації на основі наявних даних.

Некероване навчання (unsupervised learning) - це одна з основних гілок машинного навчання, де модель працює з нерозміченими даними без наявності цільових міток або відповідей.[3] (рисунок 1.2) У відміну від навчання з учителем, де маються правильні відповіді для навчання, в некерованому навчанні модель самостійно вивчає структуру та залежності в даних.

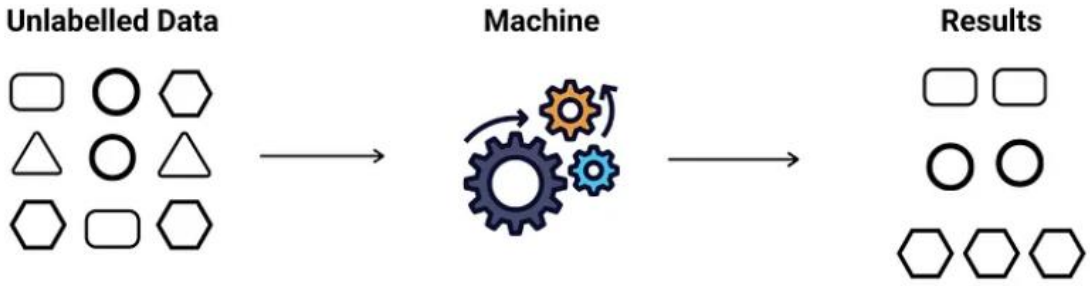


Рис 1.2. Схема роботи методу некерованого навчання.

Мета некерованого навчання полягає у виявленні прихованих закономірностей, внутрішньої структури та особливостей даних. Цей вид навчання використовується для кластеризації, виявлення аномалій, визначення патернів та зведення великого обсягу даних до більш простої та зрозумілої форми.

Наприклад, у завданні кластеризації некерована модель може групувати схожі об'єкти або елементи в певні кластери або групи, не зазначаючи конкретних категорій. У випадках виявлення аномалій, модель намагається виявити несподівані або відмінні патерни, які відрізняються від загальних структур даних.[3]

Некероване навчання застосовується в багатьох областях, таких як обробка природних мов, аналіз текстів, візуальне сприйняття, рекомендації користувачам та інші, де потрібно робити висновки на основі внутрішньої структури даних без наявності чітких цільових міток.

Навчання з підкріпленням (reinforcement learning) - це галузь машинного навчання, в якій модель навчається приймати рішення в певному оточенні з метою максимізації винагороди або досягнення певної мети. У цьому виді навчання агент взаємодіє з оточенням, спостерігаючи стан оточення та реагуючи на нього таким чином, щоб максимізувати очікувану нагороду.[4]

Головна ідея навчання з підкріпленням полягає в тому, що агент навчається шляхом спроб та помилок, діючи в певному оточенні. (рисунок 1.3.)

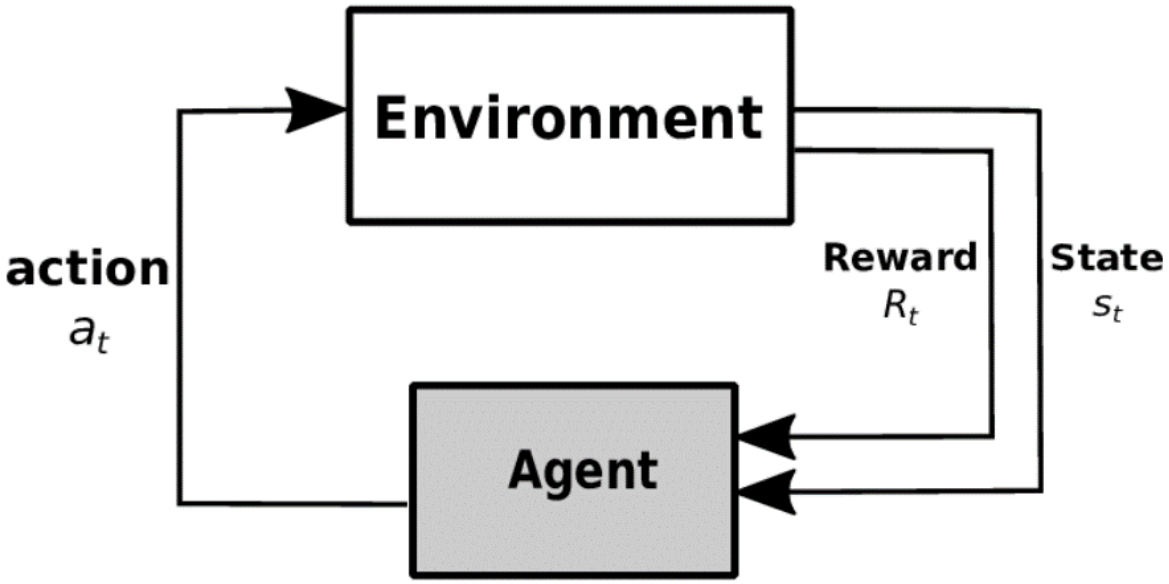


Рис 1.3. Схема роботи методу навчання з підкріпленням.

В процесі навчання агент отримує винагороду або штраф в залежності від дій, які він здійснює в певних станах оточення. Мета агента полягає в тому, щоб знайти стратегію дій, яка максимізує очікувану винагороду на протязі часу.

Цей підхід широко застосовується в областях, де необхідно приймати послідовні рішення на основі взаємодії з оточенням. Приклади застосування включають в себе робототехніку, ігрову індустрію, фінанси, управління ресурсами та інші області, де потрібно навчити систему приймати оптимальні рішення в невизначеному оточенні.

Застосування машинного навчання поширюється на багато галузей, таких як медицина, фінанси, технології, маркетинг та інші. У медицині, наприклад, воно використовується для аналізу зображень, розпізнавання хвороб та розробки нових методів діагностики. У фінансовій сфері - для прогнозування ринкових тенденцій та управління портфелем. У технологіях - для вирішення завдань, пов'язаних з розпізнаванням мови, комп'ютерним баченням і великими обсягами даних.

1.2. Аналіз технологій автономних систем транспортних засобів

Автономні транспорті засоби, які ще називають самокерованими автомобілями, розглядаються як революційна інновація 21-го століття. Хоча засновник цієї ідеї залишається невідомим, і чіткого визначення автономних транспортних засобів поки немає, термін "автономний" чітко вказує на здатність автомобіля "керувати" самостійно без участі людини.

Автономне керування, базуючись на теоріях нейронауки, ґрунтується на процесі сприйняття. Отримання оточення відноситься до збору даних від різних датчиків, таких як радар, LiDAR і Vision, та відображення ситуації за допомогою тепловізору та різних алгоритмів. Ці різні алгоритми обробки даних призводять до чіткого сприйняття та, аналізуючи різноманітні результати, дозволяють автономним транспортним засобам створювати більш повний зоровий образ навколишнього середовища[5].

З формуванням процесу сприйняття навколишнього середовища транспортного засобу можна ознайомитись на рисунку 1.4.

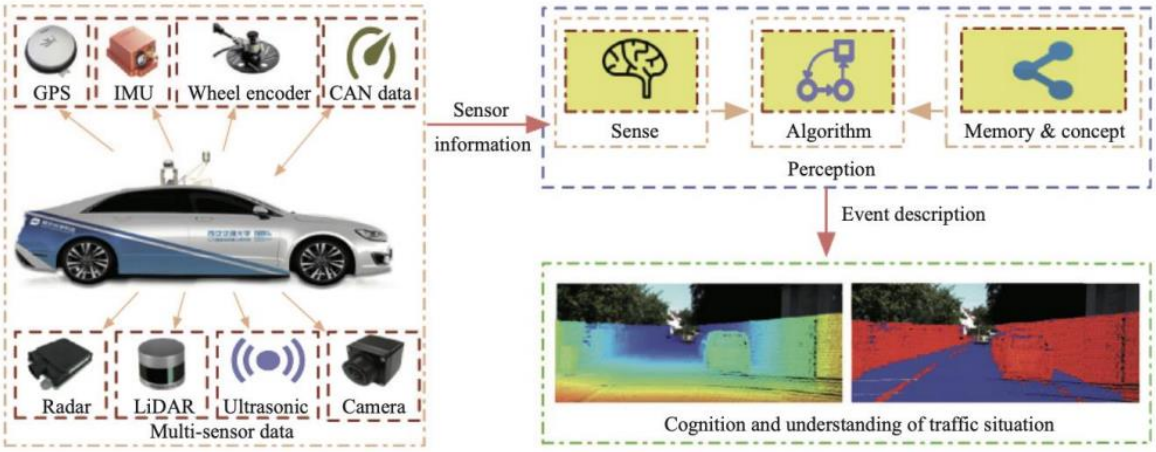


Рис 1.4. Процес формування сприйняття наволишнього середовища транспортним засобом.

Датчики транспортного засобу можна розділити на дві групи: зовнішні датчики, які вимірюють оточення, та внутрішні датчики, що записують дані самого транспортного засобу.

Multiple-Input, Multiple-Output (MIMO) - це новітня технологія радіо локації, яка застосовується у системах автономного керування. Вона має значно покращену кутову роздільну здатність, у порівнянні з попередниками. Згідно з виявленнями науковців [6], це пов'язано з автомобільними радарами, такими як MIMO, використовують частотно-модульовані безперервні сигнали (FMCW), для створення великого віртуального масиву та досягнення високої роздільної здатності як по азимуту, так і по висоті.

LIDAR (Light Detection and Ranging) - це технологія, яка використовує світло для вимірювання відстаней до об’єктів та створення детальних тривимірних зображень оточуючого середовища.[7]

З прикладом візуалізації роботи лідару в транспортному засобі можна ознайомитись на рисунку 1.5.

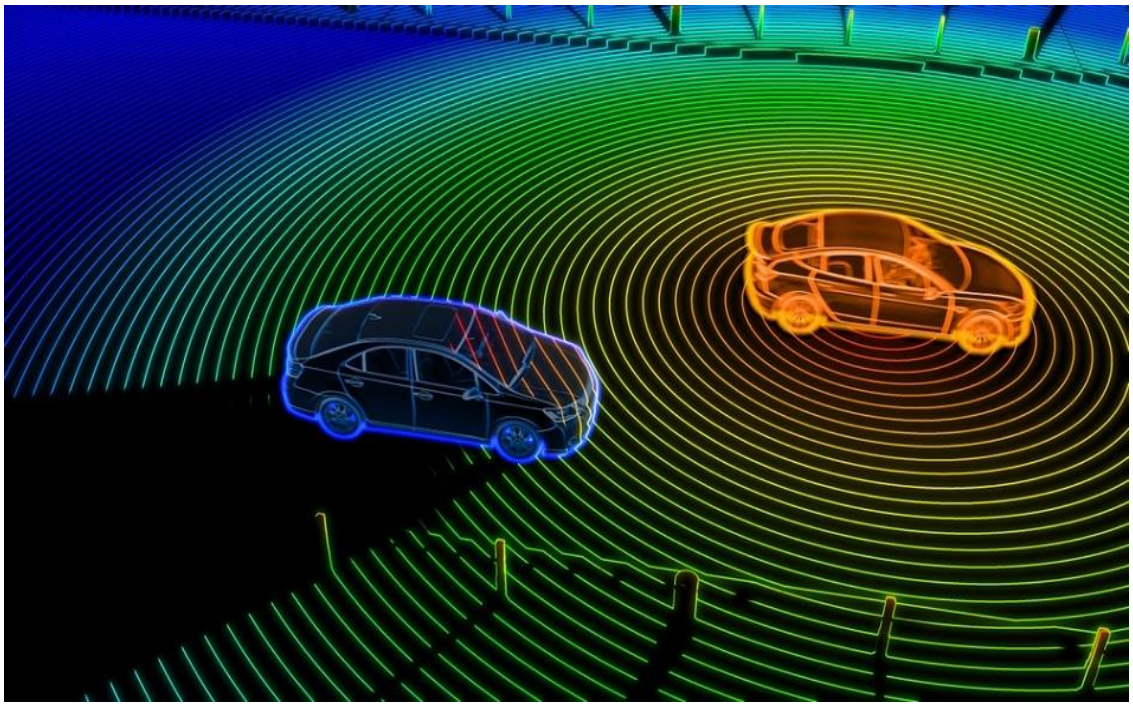


Рис 1.5. Візуалізація роботи лідару в транспортному засобі

У контексті транспортних засобів автономних систем LIDAR використовується для створення точних карт оточення, збирання даних про перешкоди та інші об’єкти, що оточують автомобіль. Ця технологія дозволяє автономним автомобілям точно визначати своє місцезнаходження, виявляти і вирізняти об’єкти, що знаходяться навколо, і приймати рішення щодо безпечної навігації по дорозі.

Система LiDAR складається з двох основних компонентів: системи лазерного далекоміра та системи сканування. Перший компонент функціонує як засіб вимірювання відстані між джерелом лазера та відбиваючою поверхнею, як це вказує сама назва. У свою чергу, система сканування обертає та керує лазерними променями під різними кутами, що також називається обертовим сканером. Раніше обертовий сканер був популярним через свою низьку вібрацію та менше навантаження на поверхню дзеркала [8]. Однак полігональні дзеркала з декількома поверхнями стали перешкодою для LiDAR через потребу високої частоти кадрів, а при обертанні вони мають значний ефект інерції через свою громіздкість, що призводить до значного енергоспоживання.

Дослідники шукали альтернативи для усунення цих недоліків, такі як дзеркала мікроелектромеханічної системи MEMS. Лідар на основі MEMS, за ствердженнями Yoo та інших [9], є більш масштабованим і діє як нелінійний генератор, що призводить до зниження робочих частот і, відповідно, до менших втрат потужності. Проста, але ефективна конструкція цих дзеркал швидко зайняла великий сегмент ринку не тільки у безпілотних автомобілях, а й у робототехніці, велосипедних системах та інших галузях високих технологій.

Безпілотні автомобілі для своєї автономної роботи використовують навігаційні системи, де GPS є важливою складовою. Типовий алгоритм такої системи може включати наступні кроки:

1) При точному визначенні поточного місцезнаходження та координат пункту призначення в певній системі координат використовуються дані з GPS та інших датчиків, що дозволяє визначити точне місце розташування автомобіля та його цільову точку.
2) Обчислення найкоротшого шляху між цими двома точками здійснюється за допомогою арифметичних операцій, шляхом врахування різних чинників, таких як дорожні умови, обмеження шляхів або швидкість руху.
3) Автомобіль направляється по визначеному шляху. Якщо виникають перешкоди на шляху, система автоматично коригує маршрут, обираючи наступний найкоротший шлях для досягнення цілі.
4) Після досягнення пункту призначення, система передає поточні координати та оновлені дані про місцеположення автомобіля для подальшого моніторингу та аналізу [10].

Останні роки принесли значний прорив у розробці програмного забезпечення для безпілотних автомобілів, особливо у поєднанні глибоких нейронних мереж із навігаційними системами. Наприклад, були проведені дослідження, [11] які привели до експериментів, які досліджували використання згорткових нейронних мереж (CNN). Було виявлено, що ці мережі проявляють надзвичайну ефективність у великих задачах сприйняття та управління безпілотними автомобілями.

Інші дослідження [12] підтверджують, що після багаторазового навчання нейронні мережі здатні передбачати кути повороту та напрямки на основі реальних даних, зібраних в реальному часі від камер або датчиків. Це дозволяє автомобілю ефективно реагувати на зміни у маршруті в реальному часі, спрощуючи процес переключення маршрутів і забезпечуючи йому продовження руху по найкоротшому шляху.

На рисунку 1.6. показано порівняння фактичних і прогнозованих кутів повороту керма, що демонструє загальний високий ступінь точності.

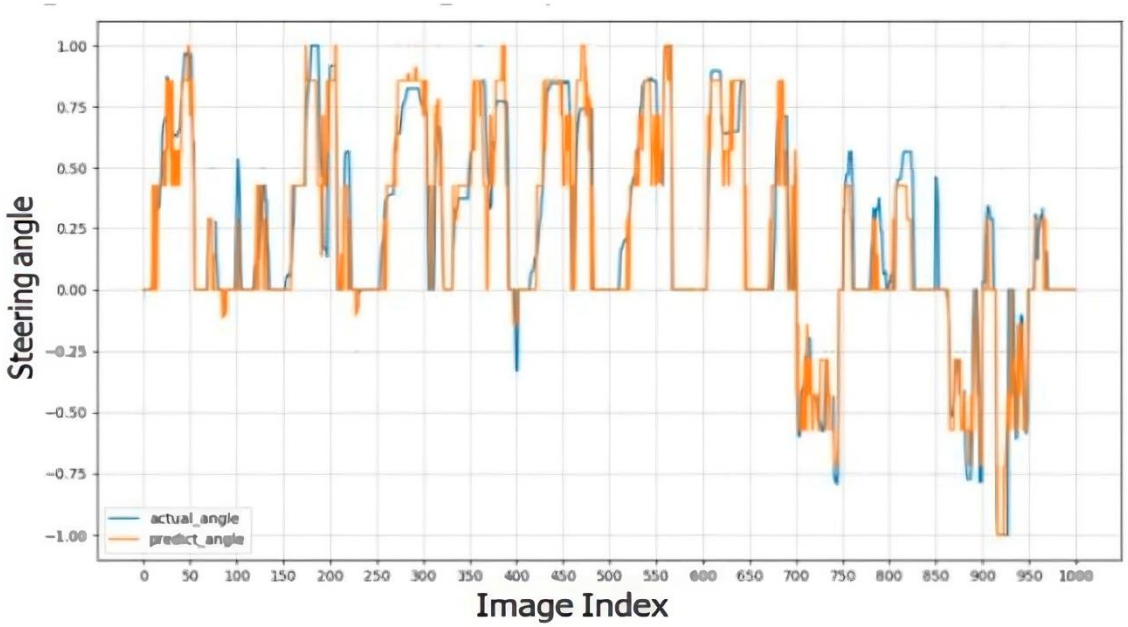


Рис 1.6. Відмінності фактичних і прогнозованих кутів повороту керма

Успіх згорткових нейронних мереж (CNN) у безпілотних автомобілях безперечно просуває дослідження у сфері автономного водіння. Навігаційні системи, які базуються на сигналах від датчиків, вносять нові виклики для систем, що залежать від сигналів, створених спеціалізованими передавачами, такими як сигнали можливостей (SOP) та глобальні навігаційні супутникові системи (GNSS).[13] Проте обидва ці типи мають спільні чотири ключові атрибути:

1) Гарантована продуктивність: Можливість людини взяти керування при необхідності.

2) Захищеність від несанкціонованого доступу: Механізми перевірки та швидке відновлення після кібер-атак.
3) Резервування: Здатність забезпечити стабільність навігаційної системи навіть при несправності датчика та виродженні сигналу.
4) Розумний рівень точності: Здатність системи забезпечити необхідний рівень точності для ефективного та надійного функціонування.

Планування шляху автономних транспортних засобів зосереджується на маршрутах руху, проте його основною увагою є поточний стан та умови самого автомобіля, наприклад, вибір маршруту, який забезпечить оптимальну швидкість та зручність для пасажирів, а не обов'язково на прямому зв'язку з пунктом призначення. Це може створити ситуацію, коли планування маршруту не відповідає командам навігаційної системи, тому важливо дотримуватись внутрішніх інструкцій для ефективної роботи. Однак, у більшості випадків ці дві системи взаємодоповнюють одна одну, забезпечуючи ефективність та безпеку. Раніше планування маршруту було складним завданням через складність прийняття рішень автономними транспортними засобами, що мають нелінійні умови управління. Однак завдяки значним дослідженням, планування шляху досягло високого рівня зрілості та успішно застосовується в промисловості автономних транспортних засобів.

В значній мірі сприяло постійному розвитку технологій планування маршруту використання двох алгоритмів, алгоритму Дейкстри та алгоритмів Беллмана-Форда. Алгоритм Дейкстри широко відомий для оптимізації маршрутів руху транспортних засобів. Однією з ключових особливостей алгоритму Дейкстри є його здатність ігнорувати всі інші непотрібні вузли після досягнення потрібного вузла, що робить його ефективним у плануванні маршруту [14].

Алгоритм Беллмана-Форда використовує концепцію релаксації зони, що означає поступове вдосконалення оцінок відстаней між вузлами в мережі. Цей алгоритм може працювати з вагованими графами, де ваги ребер показують відстань або вартість переміщення від одного вузла до іншого.

Основна ідея полягає в тому, щоб ітеративно оновлювати відстані від вихідного вузла до всіх інших вузлів у мережі. Кожна ітерація алгоритму спрямована на покращення оцінки найкоротшого шляху до кожного вузла, поступово конвергуючи до оптимального рішення.

Відмінністю алгоритму Беллмана-Форда від, наприклад, алгоритму Дейкстри, є його здатність працювати з графами, де ребра можуть мати вагу, що відповідає вартості переміщення чи часу переходу між вузлами. Це робить його більш універсальним у застосуванні до різних видів мережних структур та сценаріїв.

Однак алгоритм Беллмана-Форда динамічний протокол маршрутизації, відомі своєю повільністю реакції на зміни у мережевій топології. Це може ускладнювати адаптацію до нових умов чи різких змін у мережі, порівняно з іншими швидкими алгоритмами, які в змозі оперативно адаптуватися до таких змін[14].

Останні дослідження показують, що машинне навчання можна використовувати для прийняття рішень в автотранспорті. На основі моделювання функцій транспортного засобу та симуляції реальних умов руху, вчені розробили метод планування маршруту під назвою "Прогнозне керування моделлю" (MPC). Цей підхід було протестовано в трьох типових сценаріях: щоденне водіння по шосе, в'їзд на шосе та перетин перехрестя. Це дослідження підвищує рівень довіри та робить аналіз більш повним [15].

Результати моделювання показують, як автомобіль адаптує свою швидкість та рух відповідно до оточуючих транспортних засобів (рисунок 1.7).

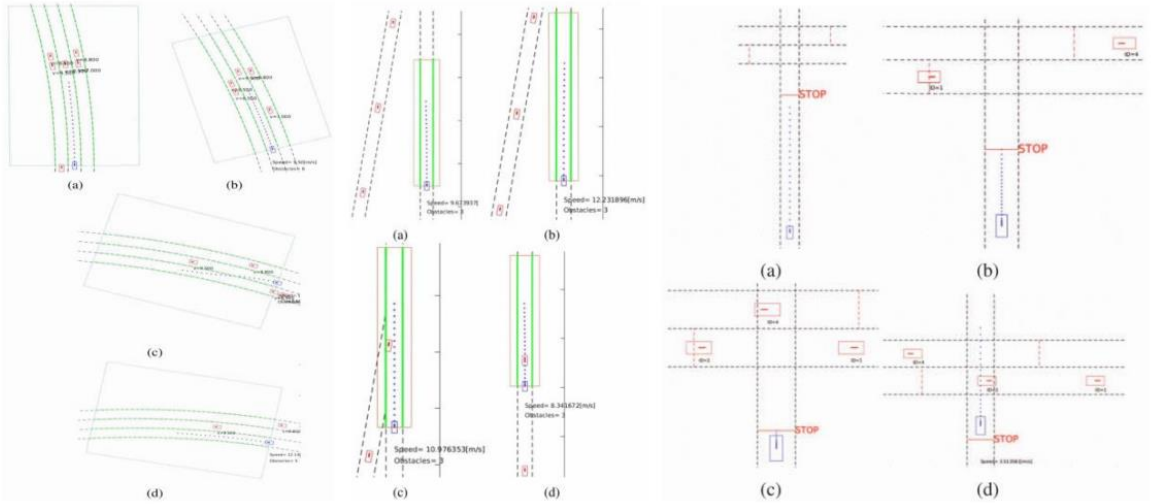


Рис 1.7. Моделювання типових сценаріїв дорожніх ситуацій

Наприклад, коли перед авто рухаються інші машини, воно знижує швидкість, навіть якщо йому потрібно їхати швидше, що відповідає людському уявленню про безпечне водіння.

Також показано, що під час зміни смуги руху автомобіль прискорюється та змінює лінію руху, забезпечуючи безпечну дистанцію від інших автомобілів. А в третьому випадку, коли підходить до знаку "стоп", він зупиняється, перетинаючи перехрестя лише у безпечних умовах.

Автономні автомобілі мають свої переваги, серед яких значна економія коштів для користувачів. Хоча вартість самого автономного автомобіля може бути вищою в порівнянні з традиційними авто через витрати на розробку технології, але після придбання він може виявитися вигідним. Це пояснюється тим, що такі авто здатні значно зменшити аварійність, особливо ті, що виникають через помилки водіїв.

Навіть з урахуванням можливих технічних неполадок, система автономного керування, яка використовує навички машинного навчання, перевершує людину в багатьох аспектах: виявленні перешкод та вправності у керуванні. Така автоматизація може дозволити зменшити ризики на дорозі та уникнути багатьох непередбачуваних ситуацій.

Автономні транспортні засоби мають потенціал сприяти різкому зменшенню нещасних випадків та летальних наслідків на дорозі. Річна економія, пов'язана з рівнем автономності, може коливатися від 17 до 350 мільярдів доларів. Окрім цього, завдяки вбудованій навігаційній системі, що автоматично обирає оптимальний маршрут, зменшується частота заторів, які часто ведуть до втрат палива [16].

Одна з причин, чому широке використання безпілотних автомобілів призводить до зменшення заторів, полягає в тому, що тепер транспортні засоби строго слідують наданим інструкціям, отриманим від точних датчиків та передових алгоритмів. Фактично, автономне водіння ефективно усуває невизначеність, спричинену різними стилями водіння людей. Це може також пояснити меншу кількість аварій, що, в свою чергу, поліпшує ефективність дорожнього руху та управління паливом.

Крім того, менша потреба в паливі для автономних транспортних засобів може допомогти зменшити негативний вплив на навколишнє середовище. Зазвичай автомобілі, що їздять по дорозі, викидають у повітря значну кількість парникових газів, таких як вуглекислий газ та інші токсичні забруднювачі, під час спалювання бензину чи дизельного палива. Викиди парникових газів, які підтверджені багатьма експериментами та дослідженнями, стали каталізатором змін клімату та забруднення атмосфери. Агентство з охорони навколишнього середовища Сполучених Штатів [17] заявляє, що звичайний легковий автомобіль викидає приблизно 4,6 метричних тон CO2 щороку, залишаючись найбільшим джерелом загальних викидів парникових газів у США на протязі багатьох років.

Натомість автономні транспортні засоби покращують ефективність використання палива та двигуна за допомогою оптимальних стратегій водіння, таких як плавне розгін та гальмування, а також режим круїз-контролю. Крім безпосереднього впливу на зменшення викидів, автономні авто можуть сприяти і використанню альтернативних джерел палива.

2 АНАЛІЗ ТА ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ

2.1 Навчання з підкріпленням та Deep Q Network

2.1.1 Навчання з підкріпленням

Навчання з підкріпленням (RL) - це галузь машинного навчання, яка вивчає, як агенти можуть навчатися приймати рішення в складних середовищах. Агент отримує винагороду за прийняття певних дій у середовищі. Агент навчається приймати рішення, які максимізують очікувану винагороду (рисунок 2.1).

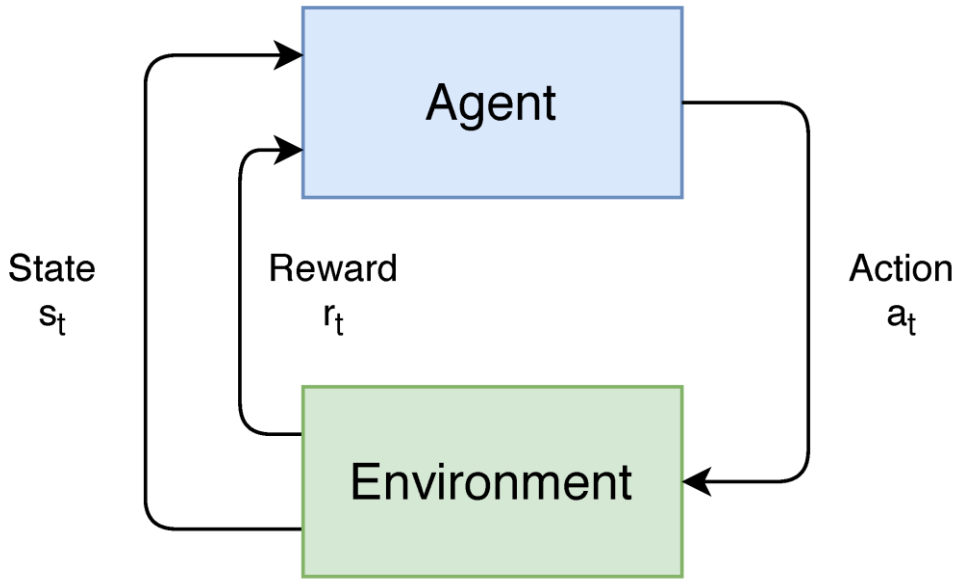


Рис. 2.1. Модель роботи навчання з підкріпленням.

Основними теоретичними основами навчання з підкріпленням є теорія ймовірності, теорія оптимального управління та теорія ігор.

Теорія ймовірності використовується для моделювання випадкових подій, які можуть відбуватися в середовищі. Наприклад, теорія ймовірності може використовуватися для моделювання того, як агент може переміщатися по середовищу або як середовище може реагувати на дії агента.

Теорія оптимального управління використовується для визначення оптимальних рішень для агентів у складних середовищах. Теорія оптимального

управління передбачає, що агент може знати всі характеристики середовища і може приймати рішення на основі повної інформації.

Теорія ігор використовується для моделювання взаємодії між агентами в середовищі. Теорія ігор може використовуватися для визначення оптимальних стратегій для агентів у ситуаціях, коли агенти взаємодіють один з одним.

Існує багато різних методів навчання з підкріпленням. Деякі з найпоширеніших методів включають:

- Q-навчання - це метод, який використовує таблицю Q для зберігання оцінки цінності кожної можливої пари (стан, дія).
- Монте-Карло - це метод, який використовує ітераційний процес для оцінки очікуваної винагороди за кожну можливу дію.
- Gradient-based policy optimization - це метод, який використовує градієнтний спуск для оновлення політики агента.

Q-навчання - це один з найпростіших і найефективніших методів навчання з підкріпленням. Q-навчання використовує таблицю Q для зберігання оцінки цінності кожної можливої пари (стан, дія). Оцінка цінності показує, скільки винагороди агент може очікувати за прийняття певної дії в певному стані.

Алгоритм Q-навчання оновлює оцінки очікуваної нагороди, використовуючи формулу Беллмана (2.1) [18]:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot \left(r_t + 1 + \gamma \cdot max_a Q(s_t + 1, a) - Q(s_t, a_t)\right) \qquad (2.1)$$

Де:

- $Q(s_t,a_t)$ - значення Q-функції для дії a_t в стані s_t на кроці t
- α - параметр, який визначає, наскільки сильно DQN буде оновлювати свої оцінки очікуваної нагороди.
- r_{t+1} - винагорода, отримана наступним кроком
- γ - фактор зниження дисконту

- $\max_a Q(s_{t+1},a)$ - максимальна оцінка очікуваної нагороди для всіх дій в наступному стані s_{t+1}

З алгоритмом роботи Q-навчання можна ознайомитись на рисунку 2.2.

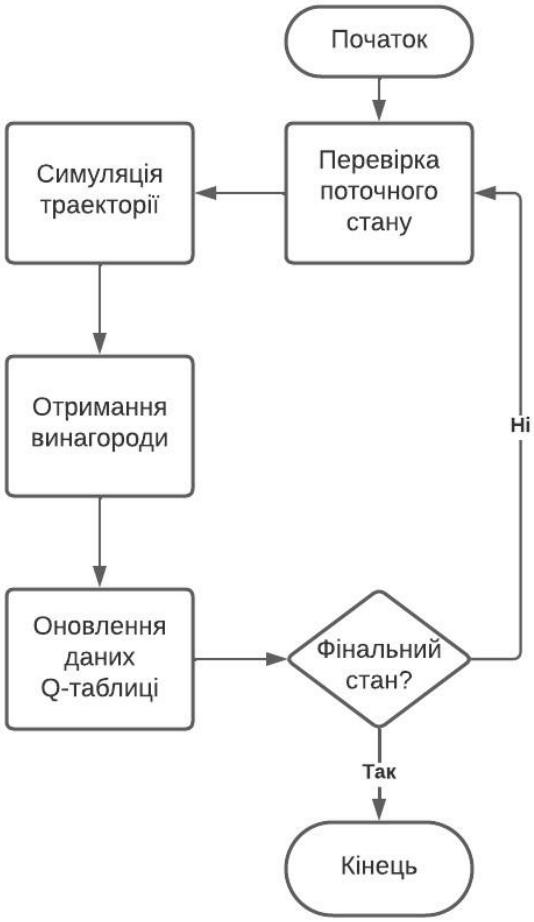


Рис 2.2. Алгоритм роботи Q-навчання

Q-навчання працює наступним чином:

1) Агент починає з випадкових дій.
2) За кожну дію агент отримує винагороду.
3) Агент оновлює таблицю Q, враховуючи отриману винагороду.
4) Агент повторює ці кроки, поки не навчиться приймати оптимальні рішення.

Монте-Карло - це метод навчання з підкріпленням, який використовує ітераційний процес для оцінки очікуваної винагороди за кожну можливу дію [19].

Монте-Карло працює наступним чином:

1) Агент починає з випадкових дій.
2) Агент отримує винагороду за кожну дію.
3) Агент оновлює оцінку очікуваної винагороди за кожну можливу дію, враховуючи отримані винагороди.
4) Агент повторює ці кроки, поки не навчиться приймати оптимальні рішення.

Основна відмінність між Q-навчанням і Монте-Карло полягає в тому, що Q-навчання використовує динамічну модель середовища, тоді як Монте-Карло не використовує. Це означає, що Q-навчання може бути ефективнішим у середовищах, де динаміка середовища добре відома. Однак Монте-Карло може бути більш ефективним у середовищах, де динаміка середовища невідома або складна.

Щоб перетворити Q-навчання на алгоритм Монте-Карло, необхідно видалити крок із динамічної моделі середовища. Це можна зробити, просто відкинувши оцінку очікуваної майбутньої нагороди в формулі Q-навчання. Це дасть нам наступну формулу (2.2) [19]:

$$Q(s,a) = Q(s,a) + \alpha\big(r + \gamma - Q(s,a)\big) \qquad (2.2)$$

Де:

- Q(s, a) - це поточне Q-значення для пари стан-дія.
- α - це коефіцієнт навчання, який визначає, наскільки сильно оновлюється Q-значення.
- r - це нагорода, яку агент отримав після виконання дії a в стані s.
- γ - це коефіцієнт дисконту, який визначає, наскільки важливі майбутні нагороди порівняно з поточними нагородами.

Цей алгоритм працює шляхом спостереження за траєкторіями в середовищі і оновлюючи Q-значення на основі цих траєкторій. Оскільки алгоритм не використовує динамічну модель середовища, він може бути повільнішим, ніж Q-навчання, особливо в середовищах з високою невизначеністю. Однак він може бути більш ефективним у середовищах, де динаміка середовища невідома або складна.

Градієнтне політичне оптимізаційне навчання (GPO)- це метод навчання з підкріпленням, який використовує градієнтний спуск для оновлення політики агента. Політика агента - це функція, яка визначає, яку дію агент повинен прийняти в кожному стані.

Градієнтне політичне оптимізаційне навчання часто використовується в контексті навчання з підкріпленням для оновлення політики агента [20]. Одним із популярних методів градієнтного політичного оптимізаційного навчання є підкріплення (2.3):

$$\theta t + 1 = \theta t + \alpha \cdot \nabla \theta J(\theta) \tag{2.3}$$

Де:

- θt - параметри політики на кроці t
- α - крок навчання
- $\nabla \theta J(\theta)$ - градієнт функції винагороди відносно параметрів політикиθ

Gradient-based policy optimization працює наступним чином:

1) Агент починає з випадкової політики.
2) Агент використовує градиентний спуск для оновлення політики, враховуючи отриману винагороду.
3) Агент повторює ці кроки, поки не навчиться приймати оптимальні рішення.

2.1.2 Deep Q Network

Deep Q Network - це тип штучного інтелекту навчання з підкріпленням, який використовується для навчання агентів приймати рішення в складних середовищах. DQN працює, навчаючи агентів прогнозувати очікувану нагороду, яку вони отримають, якщо виберуть певну дію в даному стані.

Deep Q мережі (DQN) були вперше запропоновані в 2013 році в статті DeepMind під назвою "Playing Atari with Deep Reinforcement Learning" [21]. Ця стаття була написана групою вчених під керівництвом Демиса Хассабіса, і вона описала новий метод навчання агентів приймати рішення в складних середовищах.

До появи DQN, метод підкріплення вже використовувався для навчання агентів виконувати складні завдання. Однак, попередні методи були обмежені в тому, що вони могли працювати лише з невеликою кількістю станів і дій. DQN вирішили цю проблему, використовуючи згорткові нейронні мережі для обробки інформації про стани.

Згорткові нейронні мережі є типом нейронних мереж, які призначені для обробки зображень. Вони складаються з шарів нейронів, які обробляють інформацію послідовно, починаючи з нижніх шарів, які обробляють основні характеристики зображення, і закінчуючи верхніми шарами, які обробляють складні характеристики [22].

DQN використовують згорткові нейронні мережі для обробки інформації про стани середовища. Вони навчаються прогнозувати очікувану нагороду, яку агент отримає, якщо вибере певну дію в даному стані. Це дозволяє агентам приймати оптимальні рішення в складних середовищах.

Результати дослідження DeepMind викликали великий інтерес у науковій спільноті. З того часу DQN були використані для навчання агентів виконувати широкий спектр завдань, включаючи гру в ігри, керування роботами, управління транспортними засобами та розробку стратегій.

DQN досягли значних успіхів у ряді областей. Зокрема, вони були використані для навчання агентів грати в ігри Atari на рівні людини, керувати

роботами в складних середовищах, такими як космічний простір або небезпечні промислові райони, та навіть розробляти стратегії для бізнесу та економіки. Він є потужним інструментом, який має потенціал для трансформації багатьох областей. Вони дозволяють агентам навчатися виконувати складні завдання, які раніше були неможливими.

Ось деякі конкретні приклади досягнень, які були досягнуті в результаті використання DQN:

1) У 2013 році агент, навчений на DQN, переміг людину в грі Atari Space Invaders. Гра Atari Space Invaders є класичною грою, в якій гравець керує космічним кораблем, який повинен захистити планету від інопланетних загарбників. Гра має складну ігрову динаміку, яка включає в себе управління кораблем, відстріл інопланетян і уникання їхніх снарядів. Агент отримував нагороду за знищення інопланетян і втрачав нагороду, якщо його корабель був знищений. Агент навчався прогнозувати очікувану нагороду, яку він отримає, якщо вибере певну дію в даному стані. Після навчання агент зміг перевершити результати людини в грі Space Invaders. Він міг отримувати середній бал 100,000, що набагато вище, ніж середній бал людини, який становить близько 10,000. Перемога агента, навченого на DQN, в грі Space Invaders була значним проривом в галузі штучного інтелекту. Вона показала, що агенти, навчені на DQN, можуть навчитися грати в складні ігри, які раніше вважалися недоступними для машин [21].
2) У 2016 році агент, навчений на DQN, переміг людину в грі Go. Це був значний прорив, оскільки Go вважається однією з найскладніших ігор у світі. Гра Go є древньою китайською грою, в якій гравці розміщують камені на квадратному ігровому полі. Гра має складну ігрову динаміку, яка включає в себе стратегію, позиціонування і тактику. Агент отримував нагороду за перемогу в партії і втрачав нагороду, якщо програвав. Агент був навчений на наборі даних із 30 мільярдів ігор Go. Агент, навчений на DQN, переміг людину, Лі Седоля, у п'яти партіях з шести. Це була перша перемога штучного інтелекту над людиною в грі Go [23].

3) У 2017 році безпілотний літальний апарат InSight успішно здійснив посадку на Марс. Це було перше використання DQN для керування реальним фізичним об'єктом. Безпілотний літальний апарат InSight був розроблений НАСА для вивчення геології Марса. Він був оснащений камерою, яка повинна була зняти перші зображення поверхні Марса з висоти 1,5 км. Керування безпілотним літальним апаратом InSight було складним завданням. Літальний апарат повинен був точно вирівняти свій політ і посадку, щоб уникнути зіткнення з поверхнею Марса. Для керування безпілотним літальним апаратом використовувався агент, навчений на DQN. Агент навчався на наборі даних із симуляцій посадки на Марс. Набір даних включав в себе інформацію про стан літального апарата, траєкторію його польоту та характеристики поверхні Марса. Агент навчався прогнозувати очікувану нагороду, якою була успішна посадка літального апарата на Марс. Агент успішно керував безпілотним літальним апаратом InSight, і він успішно здійснив посадку на Марс [24]. Це був значний прорив, який підтвердив, що навчені на DQN агенти, можуть бути використані для керування реальними фізичними об'єктами в складних середовищах.

Архітектура DQN складається з двох основних компонентів:

- Згорткова нейронна мережа для обробки інформації про стани середовища.
- Алгоритм Q-навчання для навчання агентів прогнозувати очікувану нагороду.

Згорткова нейронна мережа DQN складається з декількох шарів. Перший шар, або шар входу, отримує інформацію про стан середовища. Наступні шари, або шари перетворення, обробляють цю інформацію і витягують з неї важливі характеристики. Останній шар, або шар виходу, прогнозує очікувану нагороду, яку агент отримає, якщо вибере певну дію в даному стані [25].

Алгоритм Q-навчання використовується для навчання агентів прогнозувати очікувану нагороду. Алгоритм Q-навчання працює, накопичуючи дані про стани середовища, дії агентів і нагороди, які агент отримує. На основі цих даних

алгоритм Q-навчання навчає агентів прогнозувати очікувану нагороду, яку вони отримають, якщо виберуть певну дію в даному стані.

Принцип роботи DQN можна розділити на два етапи:

1) Етап оцінки. На етапі оцінки DQN отримує інформацію про стан середовища і генерує прогноз очікуваної нагороди для всіх можливих дій в цьому стані. Інформація про стан середовища подається на свертову нейронну мережу DQN. Свертова нейронна мережа використовує цю інформацію для генерування вектору характеристик, який представляє стан середовища. Вектор характеристик передається на шар повнозв'язного шару, який генерує прогноз очікуваної нагороди. Прогноз очікуваної нагороди для кожної дії представляється як число. Число більше нуля означає, що дія, ймовірно, призведе до позитивної нагороди. Число менше нуля означає, що дія, ймовірно, призведе до негативної нагороди.
2) Етап навчання. На етапі навчання DQN оновлює свої оцінки очікуваної нагороди, використовуючи дані про стани середовища, дії агентів і нагороди, які агент отримує. Дані про стани середовища, дії агентів і нагороди, які агент отримує, зберігаються в буфері пам'яті. Буфер пам'яті періодично оновлюється новими даними. На кожному кроці навчання DQN вибирає випадкове значення з буфера пам'яті. Ці значення представляють стан середовища, дію, яку агент виконав у цьому стані, і нагороду, яку агент отримав. DQN використовує ці значення для оновлення своїх оцінок очікуваної нагороди. Оновлення оцінок очікуваної нагороди виконується за допомогою алгоритму Q-навчання [26].

Стандартна формула для оновлення Q-мережі в Deep Q-Network (DQN) може бути виражена наступним чином (2.4):

$$Q(s,a) = (1-\alpha) * Q(s,a) + \alpha \cdot (r + \gamma \cdot max_a Q(s`,a`)) \quad (2.4)$$

Де:

- $Q(s,a)$ - Q-значення для дії a у стані s.

- α - швидкість навчання, яка контролює темп оновлення Q-значень.
- r - нагорода за виконання дії a у стані s.
- γ - дисконт-фактор, який визначає важливість майбутніх нагород.
- $\max Q(s',a')$ - максимальне очікуване Q-значення для наступного стану s'.

У формулі DQN для оновлення Q-мережі, крім основних компонентів формули Q-навчання, також використовується параметр $(1-\alpha)\cdot Q(s,a)$ що впливає на сам процес навчання. Це стабілізує процес оновлення Q-значень, дозволяючи моделі зберігати попередні знання, не змінюючи їх повністю при кожному оновленні. Отже, основна відмінність полягає в тому, що формула для оновлення Q-мережі в DQN додає додатковий компонент , який робить процес навчання стабільнішим та допомагає моделі зберігати попередні знання в процесі навчання.

Процес навчання DQN повторюється, поки агент не навчиться приймати оптимальні рішення в середовищі. До особливостей принципу роботи Deep Q Network можна виділити:

- DQN використовує згорткову нейронну мережу для обробки інформації про стани середовища. Це дозволяє DQN обробляти інформацію про стани середовища, представлені у вигляді зображень або інших типів даних.
- DQN використовує алгоритм Q-навчання для навчання своїх оцінок очікуваної нагороди. Алгоритм Q-навчання дозволяє DQN навчатися на основі даних про стани середовища, дії агентів і нагороди, які агент отримує.
- DQN використовує буфер пам'яті для зберігання даних про стани середовища, дії агентів і нагороди, які агент отримує. Це дозволяє DQN оновлювати свої оцінки очікуваної нагороди, використовуючи більшу кількість даних.

До переваг методу DQN можна віднести:

- Ефективність обробки інформації про стани середовища. DQN може ефективно обробляти інформацію про стани середовища, представлені у

вигляді зображень або інших типів даних. Це дозволяє DQN навчатися на великих наборах даних, які містять інформацію про різні стани середовища.

- Навчання на основі даних. DQN може навчатися на основі даних про стани середовища, дії агентів і нагороди, які агент отримує. Це дозволяє DQN навчатися з досвіду і поліпшувати свої результати з часом.
- Застосування в складних середовищах. DQN може бути використаний для навчання агентів приймати рішення в складних середовищах, таких як ігри, керування роботами та розробка стратегій.

До недоліків методу DQN можна віднести:

- Нестабільність навчання. Навчання DQN може бути нестабільним, особливо на ранніх етапах навчання. Це може призвести до того, що агент буде приймати неоптимальні рішення.
- Вимоги до ресурсів. Навчання DQN може вимагати значних ресурсів, таких як час і обчислювальна потужність.

Для вирішення основного недоліку цього методу, а саме нестабільності навчання, було обрано додаткове використання цільової мережі, та модернізація формули (2.5), шляхом розбиття Q-функції на дві частини: на базове значення та перевагу конкретної дії, що є одним із ефективних способів покращення стабільності навчання DQN.

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|A|} \cdot \sum\nolimits_{a'} A(s,a') \right) \tag{2.5}$$

Де:

- $Q(s,a)$: Q-значення для дії a у стані s.
- $V(s)$: Цінність стану s, яка представляється як базова оцінка.
- $A(s,a)$: Перевага дії a у стані s, яка визначається як різниця між Q-значенням цільової дії та середнім значенням Q-значень для усіх можливих дій у цьому стані.
- $|A|$: Кількість можливих дій у стані s.

- $1/|A| \cdot \sum A(s,a')$: Середнє значення Q-значень усіх можливих дій a' у стані s. Це значення використовується для нормалізації переваги дії $A(s,a)$.

Ця модернізована формула використовується для розрахунку Q-значення в основній мережі через розбиття на складові: цінність стану $V(s)$ та перевагу дії $A(s,a)$ відносно середнього значення переваг усіх дій у стані s. Такий підхід допомагає агентові краще оцінювати цінність стану та перевагу конкретної дії в цьому стані для вибору оптимальних дій.

Для уникнення проблеми "забування" попередніх знань та зменшення кореляції значень між навчальними прикладами, було вирішено ввести цільову функцію (2.6), що призведе до більш стабільного та швидкого навчання моделі.

$$target = r + \gamma \cdot maxQ(s', a') \tag{2.6}$$

Де:

- target - цільове Q-значення, що оцінюється для певного стану та дії.
- r - отримана нагорода за виконання дії у поточному стані.
- γ - дисконт-фактор, що визначає важливість майбутніх нагород.
- $\max Q(s',a')$ - максимальне Q-значення для наступного стану s', обчислене за допомогою самої цільової мережі.

Використання цільової мережі допомагає стабілізувати навчання, оскільки вона використовує менш корельовані дані та допомагає уникнути раптових змін у мережі.

Цільова мережа - це копія основної мережі DQN, яка оновлюється не так часто, як основна мережа. Це дозволяє основній мережі DQN навчатися, не турбуючись про те, що її оцінки очікуваної нагороди будуть змінюватися занадто швидко [27].

Механізм роботи цільової мережі такий:

1) Основна мережа DQN використовується для прийняття рішень і отримання нагород.

2) Нагороди використовуються для оновлення оцінок очікуваної нагороди основної мережі DQN.
3) Оцінки очікуваної нагороди основної мережі DQN використовуються для оновлення цільової мережі.
4) Цей процес повторюється циклічно.

Цільова мережа допомагає стабілізувати навчання DQN, оскільки вона забезпечує більш стабільне джерело для оновлення оцінок очікуваної нагороди основної мережі DQN. Це відбувається тому, що цільова мережа оновлюється не так часто, як основна мережа DQN, і тому вона менше схильна до коливань.

2.2 Огляд аналогів

2.2.1 Навчання з учителем

Навчання з учителем (supervised learning) - це парадигма машинного навчання, в якій модель навчається на наборі даних, який містить пари вхідних і вихідних даних. Модель намагається знайти функцію, яка відображає вхідні дані на вихідні дані (рисунок 2.2).

Наприклад, якщо ми хочемо навчити модель класифікувати зображення на людей і тварин, то набір даних буде містити пари зображень і їх класів. Вхідні дані для моделі будуть зображеннями, а вихідні дані будуть класами цих зображень. Функція навчання буде використовуватися для навчання моделі знаходити закономірності між зображеннями і їх класами.

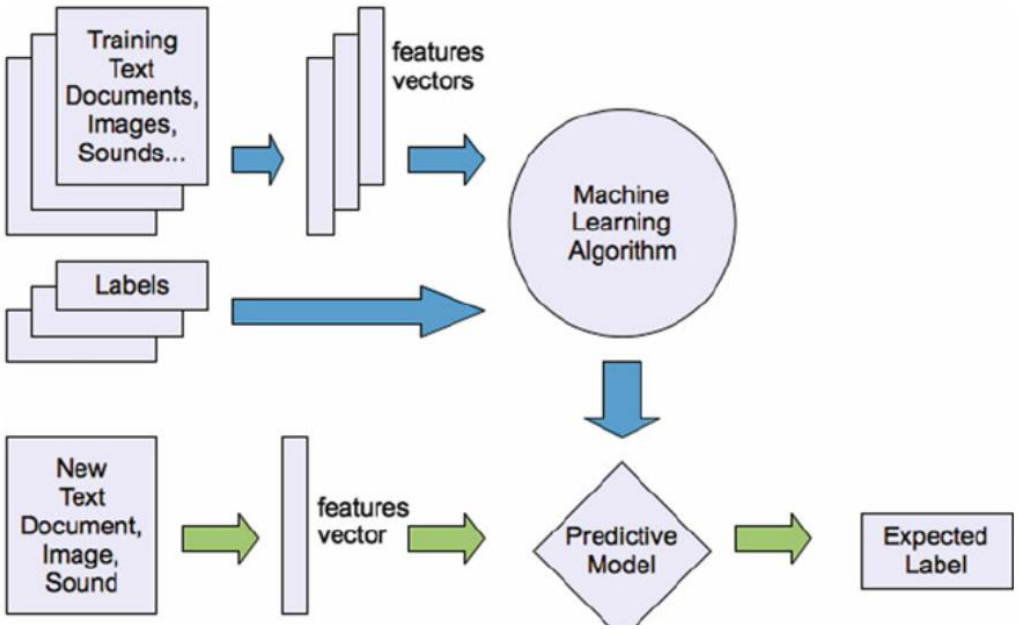


Рис 2.2. Модель роботи навчання з учителем

До основних понять навчання з учителем належать:

- Набір даних - це сукупність вхідних і вихідних даних. Вхідні дані - це дані, які подаються на модель. Вихідні дані - це дані, які модель повинна генерувати.
- Вхідні дані - це дані, які подаються на модель. Вхідні дані можуть бути представлені в різних формах, наприклад, як числа, текст, зображення або звук.
- Вихідні дані - це дані, які модель повинна генерувати. Вихідні дані також можуть бути представлені в різних формах. Наприклад, для задачі класифікації вихідні дані можуть бути категоріальними, наприклад, "квітка" або "дерево". Для задачі регресії вихідні дані можуть бути кількісними, наприклад, ціна на акції або температура.
- Функція навчання - це функція, яка використовується для навчання моделі. Функція навчання визначає, як модель буде оновлювати свої параметри в міру навчання на наборі даних.

Найбільш поширеними типами навчання з учителем є класифікація, регресія та сортування.

Класифікація - це тип навчання з учителем, в якому вихідні дані є категоріальними. Наприклад, модель може навчитися класифікувати зображення на людей і тварин.

У процесі класифікації модель навчається знаходити закономірності між вхідними даними і категоріями вихідних даних. Наприклад, модель може навчитися знаходити закономірності між характеристиками зображення і тим, чи зображено на ньому людину чи тварину.

Регресія - це тип навчання з учителем, в якому вихідні дані є кількісними. Наприклад, модель може навчитися прогнозувати ціну на акції.

У процесі регресії модель навчається знаходити закономірності між вхідними даними і кількісними характеристиками вихідних даних. Наприклад, модель може навчитися знаходити закономірності між історичними даними про ціни на акції і майбутніми цінами.

Сортування - це тип навчання з учителем, в якому вихідні дані є порядковими. Наприклад, модель може навчитися сортувати список чисел.

У процесі сортування модель навчається знаходити закономірності між вхідними даними і порядком вихідних даних. Наприклад, модель може навчитися знаходити закономірності між значеннями чисел і їх порядком у списку.

Крім трьох основних типів навчання з учителем, існують і інші типи, які менш поширені:

- Підбір - це тип навчання з учителем, в якому модель навчається підбирати параметри, які мінімізують деяку функцію втрат.
- Розподіл - це тип навчання з учителем, в якому модель навчається розподіляти ймовірність вихідних даних для даного набору вхідних даних.

Існує багато алгоритмів навчання з учителем, розглянемо одні із самих найпоширеніших.

Метод найменших квадратів - це алгоритм, який використовується для навчання моделей регресії. Метод найменших квадратів знаходить набір параметрів, який мінімізує суму квадратів помилок між фактичними й прогнозованими вихідними даними.

Лінійні моделі - це тип моделей регресії, які описуються лінійними рівняннями. Лінійні моделі можуть бути використані для вирішення широкого кола задач регресії, таких як прогнозування цін на акції, прогнозування попиту на товари та послуги тощо.

Дерева рішень - це тип моделей класифікації, які представляють знання у вигляді дерева рішень. Дерево рішень приймає рішення, розглядаючи послідовність запитань. Воно складається з вузлів, утворюючи так зване кореневе дерево, де основний вузол - корінь, не має вхідних зв'язків. Усі інші вузли мають саме один вхідний зв'язок. Вузли, які мають вихідні зв'язки, називаються внутрішніми або тестовими вузлами. Решта вузлів - це листки. У дереві прийняття рішень кожен тестовий вузол поділяє простір екземплярів на два або більше підпростори згідно з певною дискретною функцією вхідних значень. Найпростіший випадок полягає в тому, що кожен тест враховує один атрибут, розбиваючи простір екземплярів за значенням цього атрибута. Для числових атрибутів умова може описувати певний діапазон. Кожен листок відноситься до одного класу, який представляє найбільш відповідне значення цільового параметра. В листках може знаходитися вектор ймовірностей, що вказує на ймовірність певного значення цільової змінної. Екземпляри класифікуються шляхом пройдення від кореня дерева до листка відповідно до результатів тестів вздовж шляху. Кожен вузол позначений атрибутом, який перевіряється, а його гілки - відповідними значеннями [28].

На рисунку 2.3 показано просте застосування дерева прийняття рішень.

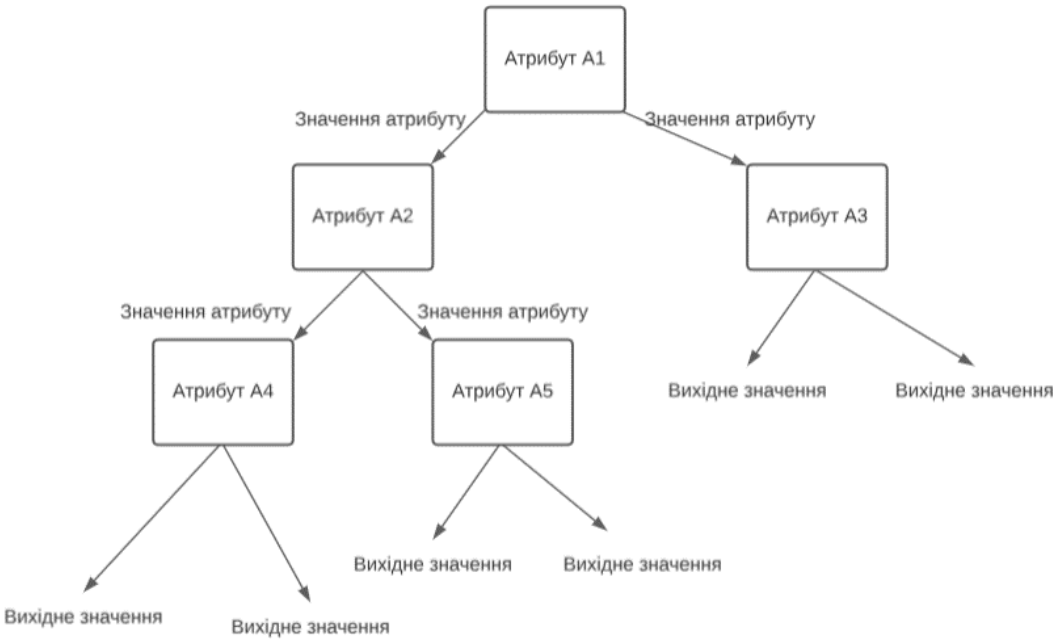


Рис 2.3. Приклад дерева прийняття рішень

Випадковий ліс - це тип моделей класифікації, який використовує ансамбль дерев рішень. Випадковий ліс генерує кілька дерев рішень (рисунок 2.4), а потім приймає рішення за допомогою голосування.

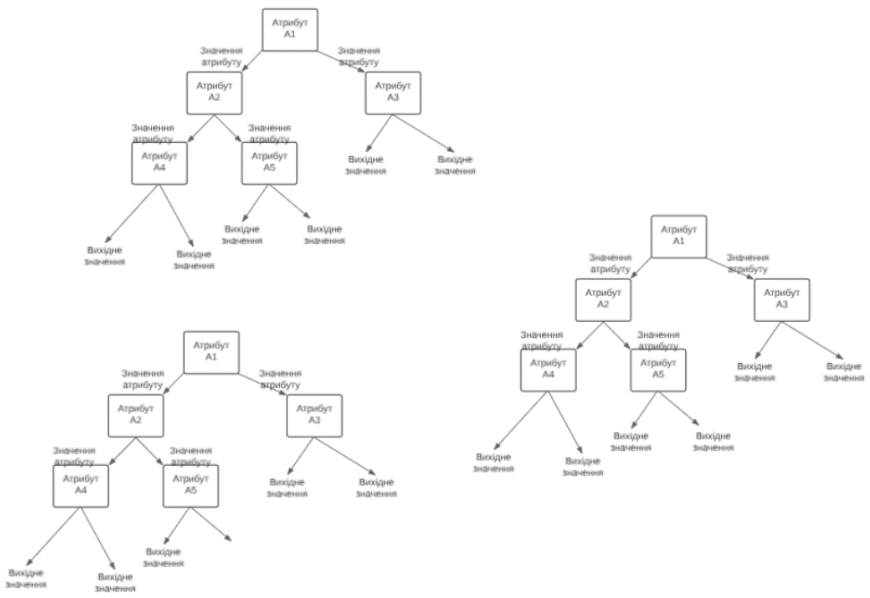


Рис. 2.4. Випадковий ліс

Нейронні мережі - це тип моделей, які можуть навчитися виконувати складні завдання, такі як розпізнавання образів і розпізнавання мови. Нейронні мережі складаються з нейронів, які з'єднані між собою. Нейрони можуть виконувати прості математичні операції, такі як додавання і множення.

До переваги навчання з учителем відносяться:

- Висока точність: моделі навчання з учителем можуть досягати високої точності в багатьох завданнях.
- Ефективність: моделі навчання з учителем можуть бути навчені швидко і ефективно.
- Застосування: навчання з учителем широко застосовується в різних областях, включаючи обробку природної мови, обробку зображень, фінанси, медицину та автоматику.

До недоліків навчання з учителем можна віднести:

- Залежність від даних: моделі навчання з учителем залежать від якості даних, на яких вони навчаються. Якщо дані неякісні, то модель може навчитися погано.
- Необхідність маркування даних: для навчання моделей навчання з учителем необхідні дані, які вже містять правильні відповіді. Маркування даних може бути трудомістким і дорогим процесом.

2.2.2 Навчання без учителя

Навчання без учителя - це парадигма машинного навчання, в якій алгоритми навчаються на наборі даних, який не містить мічених даних. Мічені дані - це дані, які містять правильні відповіді на завдання, яке має бути вирішено.

Навчання без учителя може бути використано для вирішення широкого кола задач, включаючи:

- Кластеризація - це завдання групування даних у схожі групи. Навчання без учителя може бути використано для кластеризації даних на основі їхніх спільних характеристик. Наприклад, навчання без учителя може бути

використано для групування покупців у схожі групи на основі їхніх покупок. Вона може бути використана для групування покупців у схожі групи на основі їхніх покупок. Наприклад, компанія, яка продає товари і послуги онлайн, може використовувати навчання без учителя для групування своїх клієнтів за їхніми покупками. Це може допомогти компанії зрозуміти, які типи продуктів і послуг цікавлять різні групи клієнтів.

- Різнасність- це завдання виявлення аномалій у даних. Навчання без учителя може бути використано для виявлення аномалій у даних, які відрізняються від більшості інших даних. Наприклад, навчання без учителя може бути використана для виявлення шахрайських транзакцій у банківських записах. Наприклад, банк може використовувати навчання без учителя для виявлення транзакцій, які відрізняються від більшості інших транзакцій. Це може допомогти банку запобігти шахрайству.
- Асоціація - це завдання виявлення закономірностей між змінними. Вона може бути використана для виявлення закономірностей у продажах товарів і послуг. Наприклад, магазин може використовувати навчання без учителя для виявлення закономірностей у тому, які товари і послуги часто купуються разом. Це може допомогти магазину розміщувати товари в магазині таким чином, щоб заохочувати клієнтів до купівлі більшого числа товарів.

Навчання без учителя може бути використано для виявлення закономірностей між змінними, які не можуть бути легко виявлені за допомогою інших методів. Наприклад, навчання без учителя може бути використано для виявлення закономірностей у продажах товарів і послуг.

Навчання без учителя має ряд переваг порівняно з навчанням з учителем. Воно не вимагає мічених даних, які можуть бути трудомісткими і дорогими для отримання. Крім того, навчання без учителя може бути більш надійним у випадку, коли дані є неповними або шумними. Однак, навчання без учителя також має ряд

недоліків. Воно може бути менш точним, ніж навчання з учителем, і може бути більш складним для реалізації.

Навчання без учителя - це потужний інструмент, який може бути використаний для вирішення широкого кола задач. Воно має ряд переваг порівняно з навчанням з учителем, але також має ряд недоліків. Вибір того, який підхід до навчання машинного навчання використовувати, залежить від конкретних завдань, які необхідно вирішити.

3 РЕАЛІЗАЦІЯ МЕТОДУ АВТОМАТИЗОВАНОГО КЕРУВАННЯ АВТОТРАНСПОРТОМ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Аналіз сучасних засобів програмної інженерії для реалізації методу автоматизованого керування автотранспортом

3.1.1 TensorFlow

TensorFlow - це бібліотека з відкритим кодом для чисельних обчислень та глибокого навчання, розроблена командою Google Brain у 2015 році. Вона об'єднує різноманітні моделі та алгоритми машинного навчання, надаючи зручний програмний інтерфейс для їх використання [29]. TensorFlow використовує Python або JavaScript для створення додатків і має можливість виконувати їх на високопродуктивній C++.

У порівнянні з PyTorch та Apache MXNet, TensorFlow може навчати та застосовувати глибокі нейронні мережі для різних завдань, включаючи класифікацію рукописних цифр, розпізнавання зображень та роботу з текстом та переклад мови. Важливими перевагами є можливість масштабного прогнозування виробництва та наявність багатої бібліотеки попередньо навчених моделей для використання у власних проектах. TensorFlow також надає приклади найкращих практик з використання коду з колекції моделей, які можна використовувати для навчання власних моделей.

Суть його роботи полягає в тому, що він дозволяє створювати графи потоків даних, які моделюють переміщення даних через вузли обробки. Кожен вузол представляє математичну операцію, а з'єднання між вузлами - це тензори, або багатовимірні масиви даних.

Програми на TensorFlow можна запускати на різних платформах, включаючи локальні комп'ютери, хмарні середовища, мобільні пристрої та спеціалізовані модулі, наприклад, модуль обробки TensorFlow (TPU) Google для

прискорення [30]. Отримані моделі можна використовувати для прогнозування на різних пристроях.

У версії TensorFlow 2.0, випущеній у жовтні 2019 року, були внесені значні поліпшення для зручності користувачів. Наприклад, API Keras став доступним для простого навчання моделей, а підтримка TensorFlow Lite спрощує розгортання моделей на різних платформах. Також у версії 2.0 полегшено розподілене навчання та використання моделей.

Навчені моделі можуть бути використані для прогнозування через контейнери Docker, використовуючи API REST або gRPC. Для більш складних сценаріїв обслуговування можна використовувати Kubernetes.

TensorFlow пропонує різноманітні можливості для розробників, використовуючи мову програмування Python. Python є простим у вивченні та має зручний синтаксис, і використання його дозволяє зручно виражати високорівневі абстракції. TensorFlow підтримує версії Python від 3.7 до 3.10, хоча може працювати із попередніми версіями, але це не завжди гарантується.

У TensorFlow вузли та тензори представлені як об'єкти Python, але самі обчислення не виконуються в мові Python. Реалізація математичних операцій виконується у високопродуктивних бібліотеках, які написані на C++. Python використовується для керування потоком даних та надає високорівневі можливості програмування[31].

Користування TensorFlow на високому рівні, таке як створення вузлів та шарів і зв'язування їх, використовує бібліотеку Keras. Keras API простий для використання зовнішньо: наприклад, базову модель із трьома шарами можна створити менше, ніж за 10 рядків коду, а код для її навчання — ще кілька рядків. Але якщо потрібно здійснити більш тонку роботу, наприклад, створити власний цикл тренування, то це також не викликає ніяких проблем.

Python є основною мовою для TensorFlow і машинного навчання загалом, але JavaScript тепер також визнаний як ключовий інструмент для роботи з TensorFlow. Його велика перевага полягає в тому, що він може працювати в будь-якому веб-браузері.

TensorFlow.js, бібліотека TensorFlow для JavaScript, яка використовує WebGL API для прискорення обчислень за допомогою графічних процесорів, що доступні в системі. Також можна використовувати WebAssembly для виконання, що прискорює обчислення, особливо на центральних процесорах. Однак краще використовувати графічні процесори, коли це можливо. Ви можете почати з вже підготовленими моделями, що дозволить вам швидко розпочати роботу з простими проектами та отримати уявлення про їхню роботу.

Навчені моделі TensorFlow можна впроваджувати на периферійних комп'ютерах та мобільних пристроях, таких як iOS або Android. Набір інструментів TensorFlow Lite оптимізує моделі TensorFlow для успішної роботи на цих пристроях, дозволяючи знайти баланс між розміром моделі та її точністю. Менші моделі можуть бути менш точними, але втрата точності зазвичай невелика і може бути компенсована швидкодією та енергоефективністю моделі.

TensorFlow конкурує з безліччю інших фреймворків у світі машинного навчання. PyTorch, CNTK і MXNet є основними альтернативами, які задовольняють подібні потреби.

- PyTorch також створено на Python і має багато спільного з TensorFlow: внутрішні компоненти для апаратного прискорення, інтерактивну модель розробки та корисні вбудовані компоненти. PyTorch часто вибирають для швидкої розробки проектів, які потрібно відразу запустити, тоді як TensorFlow краще підходить для великих та складних проектів. Тобто, PyTorch — це оптимальний вибір для зручного старту, а TensorFlow — для глибоких та великих за обсягом проектів.
- CNTK, або Microsoft Cognitive Toolkit, подібний до TensorFlow у використанні графової структури для опису потоку даних, проте більше спрямований на розробку глибоких нейронних мереж. Його переваги — в швидкості виконання багатьох завдань нейронних мереж та розширеному наборі API: Python, C++, C#, Java. Однак в освоєнні та розгортанні CNTK він наразі не такий простий, як TensorFlow. Важливо відзначити, що CNTK

доступний лише за ліцензією GNU GPL 3.0, в той час як TensorFlow має більш ліберальну ліцензію Apache. Крім того, розвиток CNTK не настільки активний, останнє оновлення якого відбулося у 2019 році.

- Apache MXNet, обраний Amazon як основний фреймворк глибокого навчання на AWS, може масштабуватися майже лінійно на кількох графічних процесорах та машинах. Він підтримує різноманітні мовні API: Python, C++, Scala, R, JavaScript, Julia, Perl, Go. Однак робота з його власними API не така зручна, як з TensorFlow, та має меншу кількість користувачів та розробників.

Найбільшою перевагою TensorFlow у машинному навчанні є його рівень абстракції. Замість праці з дрібними деталями алгоритмів чи з'ясуванням способів зв'язку вхідних та вихідних даних, розробник може сконцентруватися на загальній логіці програми. TensorFlow самостійно керує дрібницями.

Платформа надає інструменти для полегшення розробки, нагляду та самоаналізу додатків, побудованих на ній. Операції з графіком можна оцінювати та змінювати окремо, а "режим активного виконання", який раніше був опцією, тепер став стандартним.

TensorBoard дозволяє контролювати та профілювати роботу графіків у веб-панелі. Сервіс Tensorboard.dev, розміщений Google, дозволяє розміщувати та обмінюватися експериментами у машинному навчанні. Він є безкоштовним для користувачів з обмеженнями на сховище обсягом до 100 мільйонів скалярів, 1 ГБ тензорних даних і 1 ГБ бінарних об'єктів. Треба мати на увазі, що дані, розміщені в Tensorboard.dev, стають загальнодоступними.

Завдяки підтримці від Google, TensorFlow має чимало переваг. Компанія допомагає активно розвивати проект і надає корисні пропозиції, що спрощують процес розгортання та використання платформи. Наприклад, впровадження кремнієвих TPU в хмару Google є лише одним із багатьох прикладів.

3.1.2 NumPy

NumPy є ключовою бібліотекою для роботи з масивами в мові програмування Python. Вона відіграє важливу роль в різноманітних дослідженнях у таких галузях, як фізика, хімія, астрономія, геологія, біологія, психологія, матеріалознавство, інженерія, фінанси та економіка.

Наприклад, в астрономії NumPy використовується для виявлення гравітаційних хвиль та створення першого зображення чорної діри. У цьому контексті розглядаються основні концепції масиву, які допомагають створити просту та потужну парадигму програмування для обробки та аналізу наукових даних.

NumPy є основою наукової екосистеми Python і діє як важливий компонент для взаємодії з іншими бібліотеками обчислень масивів. Його широке використання призвело до розробки інтерфейсів та масивів, схожих на NumPy, у декількох спеціалізованих проектах. З центральним положенням в екосистемі, NumPy виступає як ключовий засіб для підтримки наукового та промислового аналізу у наступному десятилітті.

Два пакети масивів для Python існували перед NumPy. Пакет Numeric був створений у середині 1990-х років і надавав об'єкти-масиви та функції для роботи з масивами у Python. Він був написаний мовою C і включав швидкі реалізації лінійної алгебри. Спочатку він був використаний для управління програмами C++ у дослідженнях термоядерного синтезу в Ліверморській національній лабораторії.

Для роботи з великими астрономічними зображеннями від космічного телескопа Хаббл, була розроблена оновлена версія Numeric, відома як Numarray. Вона включала покращену підтримку структурованих масивів, гнучке індексування, відображення пам'яті та інші покращення.

Хоча Numarray був сумісний із Numeric, у 2005 році з'явився NumPy як об'єднання кращих функцій обох пакетів. Він поєднав функціональність Numarray з продуктивністю Numeric та його потужним C API. Це об'єднання стало успішним і визнано як "найкраще з обох світів".

NumPy є основою практично кожної бібліотеки Python, спрямованої на виконання наукових чи числових обчислень, включаючи SciPy, Matplotlib, та pandas. Ця відкрита бібліотека, розроблена спільнотою, надає багатовимірний об'єкт Python-масив разом із різноманітними функціями для роботи з масивами. Благодаря своїй простоті, масив NumPy став фактичним стандартом для обміну даними у форматі масиву в середовищі Python. Це визначеною мірою сприяє інтеграції NumPy з іншими бібліотеками, такими як scikit-image, scikit-learn, і багатьма іншими.

NumPy оптимізований для роботи з масивами в пам'яті за допомогою центрального процесора (CPU). З огляду на сучасні тенденції у сфері апаратного забезпечення, з'явилося безліч нових пакетів масивів для Python, спрямованих на використання спеціалізованих сховищ та апаратного забезпечення.

Відмінною рисою цих нових бібліотек є їхня спеціалізованість, але одночасно вони стикаються з викликом об'єднати користувачів, оскільки вже існуюча спільнота побудована на NumPy. Хоча поділ Numarray та Numeric мав свої нюанси, нові бібліотеки важче втручаються у вже існуючу екосистему.

З метою забезпечення спільноті доступу до новітніх технологій, NumPy перетворюється на централізований координуючий механізм. Він визначає чітко визначений API програмування масивів та відправляє його до спеціалізованих реалізацій масиву за необхідності.

Масив NumPy – це структура даних, яка ефективно зберігає багатовимірні масиви, відомі також як тензори, і надає зручний доступ до них. Він складається з вказівника на пам'ять, а також метаданих, що використовуються для інтерпретації збережених даних. Серед цих метаданих важливі такі параметри, як "тип даних", "форма" та "кроки". Масив NumPy забезпечує широкі можливості для виконання наукових обчислень в Python.

Тип даних визначає природу елементів, що зберігаються в масиві. Кожен масив має один і той же тип даних, і розмір кожного елемента визначається в однаковій кількості байтів у пам'яті. У масиві NumPy кожен елемент має однаковий тип даних. Приклади типів даних включають дійсні та комплексні

числа різної точності, рядки, мітки часу та покажчики на об'єкти Python. Обрання правильного типу даних важливо для оптимізації використання пам'яті та виконання операцій.

Форма масиву визначає кількість елементів уздовж кожної осі, і розмірність масиву визначається кількістю його осей. Наприклад, вектор чисел можна зберегти як одновимірний масив з формою (n,), де n - кількість елементів. У той час як кольорове відео може бути представлене як чотиривимірний масив, де форма вказує кількість кадрів, висоту, ширину та кількість кольорів.
Наприклад, якщо відео має розмірність 1280x720 пікселів, 24 кадри на секунду і використовується 3 канали для кодування кольору (RGB), то його форма може бути (24, 720, 1280, 3).

Для інтерпретації комп'ютерної пам'яті, яка лінійно зберігає елементи, як багатовимірні масиви, потрібні відповідні кроки. Ці кроки визначають, скільки байтів необхідно перемістити в пам'яті для переходу між рядками, стовпцями та іншими елементами. Наприклад, для двовимірного масиву чисел із плаваючою комою форми (4, 3), де кожен елемент займає 8 байт у пам'яті, кроки будуть дорівнювати (24, 8).

NumPy може зберігати масиви в порядку пам'яті C або Fortran, що визначає порядок зберігання даних. У порядку C елементи зберігаються вдовж рядків, тоді як у порядку Fortran - вдовж стовпців. Це дозволяє зовнішнім бібліотекам, написаним на цих мовах, прямо отримувати доступ до даних масиву NumPy у пам'яті.

Користувачі взаємодіють з масивами NumPy через процес "індексування" для доступу до підмасивів або окремих елементів. Крім того, вони використовують оператори (наприклад, +, − і ×) для векторизованих операцій та @ для множення матриць. Функції, що підтримують масиви, також входять у взаємодію з масивами, створюючи легко читаний, виразний та високорівневий API для програмування масивів. Усе це дозволяє NumPy ефективно керувати базовою механікою швидкого виконання операцій.

Процес індексації масиву дозволяє отримати окремі елементи, підмасиви або елементи, що задовольняють певну умову. Навіть інші масиви можуть служити індексами для індексації. Там, де це можливо, індексація, що повертає підмасив, створює "перегляд" вихідного масиву, що дозволяє ефективно працювати з підмножинами даних та економити використання пам'яті.

Щоб розширити синтаксис масиву, NumPy включає функції, які виконують векторизовані обчислення масивів, такі як арифметика, статистика та тригонометрія.

Векторизація - це обробка цілих масивів, а не їх окремих елементів, що є ключовою для програмування масивів. Це означає, що операції, які вимагають багато рядків у мовах, таких як C, часто можна виразити як єдиний зрозумілий вираз у Python. Такий підхід дозволяє створювати компактний код і дозволяє користувачам зосередитися на деталях аналізу, тоді як NumPy ефективно обробляє цикли над елементами масиву, використовуючи швидку кеш-пам'ять комп'ютера, оптимізуючи їхню обробку.

Під час виконання векторизованої операції, наприклад, додавання, над двома масивами однакової форми логічно припускати, що має відбутися. Завдяки трансляції, NumPy може працювати над масивами з різними розмірами, забезпечуючи результати, які легко сприймаються. Простий приклад - додавання скалярного значення до масиву. Але трансляція також розширюється на більш складні сценарії, такі як масштабування кожного стовпця масиву чи генерація сітки координат. Під час трансляції один або обидва масиви фактично дублюються без копіювання даних у пам'яті, щоб форми операндів збігалися. Трансляція також застосовується, коли масив індексується за допомогою масивів індексів.

Інші функції, які підтримують масив, такі як сума, середнє значення та максимум, виконують поелементне "зменшення", агрегуючи результати по одній, кількох або всіх осях масиву. Наприклад, підсумовування n-вимірного масиву за d осями призводить до масиву розмірністю n – d.

NumPy також включає функції з підтримкою масивів для створення, зміни форми, конкатенації та доповнення масивів; пошуку, сортування та підрахунку даних; а також читання та запису файлів. Він надає розширену підтримку для генерування псевдовипадкових чисел, містить ряд розподілів ймовірностей і виконує прискорену лінійну алгебру, використовуючи одну з кількох оптимізованих серверних бібліотек, таких як OpenBLAS18, 19 або Intel MKL, залежно від доступного обладнання.

Комбінація простого представлення масиву в пам'яті, синтаксису, що точно відтворює математичні операції, та різноманітних допоміжних функцій, які підтримують масиви, утворює результативну та потужну мову програмування для ефективної роботи з масивами.

Python – це мова програмування загального призначення з відкритим вихідним кодом, яка інтерпретується. Вона ідеально підходить для вирішення звичайних задач програмування, таких як обробка даних, взаємодія з веб-ресурсами та аналіз тексту. Додавання швидких операцій із масивами та лінійної алгебри дозволяє вченим виконувати всю свою роботу в рамках однієї мови програмування. Python особливо популярний в академічному середовищі, оскільки його вважають легким для вивчення та викладання, і він широко використовується як основна мова навчання в університетах

Навіть не будучи складовою стандартної бібліотеки Python, NumPy володіє важливими перевагами завдяки тісним взаєминам з розробниками Python. З течією часу мова Python додала нові можливості та спеціальний синтаксис для того, щоб робота з NumPy була більшою мірою зручною і легко читається. Проте через те, що NumPy не є частиною стандартної бібліотеки, він зберігає свою автономію в плані політики випуску та шаблонів розробки.

SciPy і Matplotlib тісно взаємопов'язані з NumPy з точки зору історії розвитку та використання. SciPy надає фундаментальні алгоритми для наукових обчислень, включаючи математичні, наукові та інженерні процедури. Matplotlib генерує графіки та візуалізації, готові для публікації. У поєднанні з NumPy, SciPy

і Matplotlib, а також розширеним інтерактивним середовищем, таким як IPython або Jupyter, вони створюють потужну інфраструктуру для обчислень.

NumPy, що є основою для екосистеми бібліотек з підтримкою масивів, встановлює стандарти документації, забезпечує інфраструктуру тестування масивів і додає підтримку збірки для Fortran та інших компіляторів. SciPy та Matplotlib, які базуються на цьому фундаменті, надають кілька широко використовуваних бібліотек для наукових обчислень та візуалізації. Наукова екосистема Python з цими бібліотеками є потужним інструментом для різноманітних обчислювальних задач.

Багато наукових дослідницьких груп розробляють значні наукові бібліотеки, які додають функціональні можливості до екосистеми Python. Наприклад, бібліотека eht-imaging, створена спільнотою Event Horizon Telescope для обробки радіоінтерферометричних зображень та моделювання, широко використовує масиви NumPy для зберігання та обробки числових даних на всіх етапах обробки даних.

SciPy надає інструменти для різноманітних завдань обробки зображень, включаючи фільтрацію та вирівнювання, а бібліотека scikit-image розширює ці можливості, надаючи більш високорівневі функції, такі як граничні фільтри та перетворення Хафа.

Додатково, модуль scipy.optimize використовується для математичної оптимізації, NetworkX для комплексного аналізу мережі, а Astropy для обробки стандартних астрономічних форматів файлів та обчислення координат часу перетворення. Matplotlib використовується для візуалізації даних та створення остаточного зображення чорної діри. Ці бібліотеки разом утворюють потужний інструментарій для вчених у різних галузях.

Інтерактивне середовище, яке базується на програмуванні масивів та використовує навколишню екосистему інструментів, особливо у сфері аналізу даних, ідеально підходить для дослідницької роботи. В основі цього середовища зазвичай лежать IPython або Jupyter, де користувачі можуть ефективно

взаємодіяти зі своїми даними, виконувати маніпуляції та візуалізацію, а також швидко ітерувати для вдосконалення коду.

Користувачі можуть вести експерименти, перевіряти гіпотези та проводити аналіз, підглядаючи результати крок за кроком. Це дозволяє їм ефективно взаємодіяти з даними та висвітлювати важливі відомості в процесі дослідження. Окрім того, ці ітеративні сеанси можуть бути об'єднані в імперативні або функціональні програми або зошити, що містять як обчислення, так і розповідь про результати.

Для роботи з науковими обчисленнями, окрім інтерактивних середовищ, де можна проводити дослідницьку роботу, використовуються інші інструменти, такі як текстові редактори або інтегровані середовища розробки (IDE), наприклад, Spyder. Ці інструменти допомагають науковцям створювати, вдосконалювати та виконувати складні обчислення.

Культура розробки програмного забезпечення в середовищі NumPy є ключовим чинником для сприяння дослідницької роботи та швидкого створення прототипів. Для підвищення співпраці та зменшення помилок, команда NumPy впровадила ряд ефективних практик розробки програмного забезпечення.

Однією з цих практик є використання перевірених часом методів розробки програмного забезпечення. Команда дотримується високих стандартів, які були розглянуті та прийняті не лише керівниками проекту, але й з ентузіазмом навчають новачків, щоб розширювати спільноту.

Додатково, NumPy використовує розподілений контроль версій і систему перегляду коду. Це допомагає покращити співпрацю над кодом, забезпечуючи ефективний обмін ідей та виправлення помилок. Безперервне тестування є ще однією важливою складовою, де для кожної запропонованої зміни в NumPy автоматизовано запускається велика кількість тестів для забезпечення стабільності та надійності коду.

Проект також славиться своєю повною, високоякісною документацією, інтегрованою з вихідним кодом. Це допомагає розробникам та користувачам швидше розуміти функціоналість бібліотеки та використовувати її ефективно.

Ця культура використання найкращих практик для розробки наукового програмного забезпечення визначає екосистему бібліотек, які будуються на основі NumPy. Прикладом успішної імплементації цієї культури є проект Astropy, який отримав визнання від Королівського астрономічного товариства.

У нагороді, присудженій Astropy, зазначається, що проект надав сотням молодих науковців досвід професійних стандартів розробки програмного забезпечення. Це включає в себе використання контролю версій, модульного тестування, перевірку коду та процедури відстеження проблем. Такий досвід є життєво важливим для сучасних дослідників і часто відсутній у формальній університетській освіті з фізики чи астрономії.

Члени спільноти активно працюють над усуненням цього недоліку шляхом надання курсів і семінарів, щоб допомогти науковцям отримати необхідні навички у розробці програмного забезпечення. Таким чином, вони сприяють розвитку та зміцненню культури найкращих практик в науковому програмуванні.

Нещодавнє стрімке зростання наукових областей, таких як наука про дані, машинне навчання та штучний інтелект, значно підсилило використання Python у наукових дослідженнях. Застосування цієї мови програмування важливіше та різноманітніше, від бібліотеки зображень , до різних областей природничих і соціальних наук. Вони стали необхідними інструментами в багатьох галузях, використовуючи NumPy та його екосистему.

NumPy та його екосистему вивчають на університетських курсах, навчальних таборах і літніх школах. Крім того, вони є предметом громадських конференцій і семінарів по всьому світу. API NumPy стало стандартом в галузі наукового програмування, що відображається в його всеосяжному використанні та широкому розповсюдженні.

NumPy надає можливість працювати з багатовимірними однорідно типізованими масивами в пам'яті на центральному процесорі (ЦП). Це дозволяє використовувати масиви на різних пристроях, від вбудованих систем до великих суперкомп'ютерів. Важливо відзначити, що продуктивність NumPy наближається

до тієї, яку можна отримати при використанні скомпільованих мов програмування.

Протягом більшості свого існування NumPy в основному використовувався для обчислень, пов'язаних з масивами, і відігравав ключову роль у великій кількості наукових і технічних областей, де важлива ефективність та швидкодія.

З часом виникла проблема розриву між зростанням обсягів наукових даних і сучасними апаратними архітектурами, зокрема в контексті потреб глибокого навчання та штучного інтелекту, де використання спеціалізованих апаратних прискорювачів, таких як GPU, TPU і FPGA, стає все більш поширеним.

NumPy, як програмна бібліотека, має свої обмеження у використанні розподілених даних та прискорювачів. Проте існують інші інструменти та бібліотеки, які створені з урахуванням цих викликів. Наприклад, Dask та Apache Spark дозволяють працювати з розподіленими даними, а TensorFlow та PyTorch надають можливість використовувати апаратні прискорювачі для глибокого навчання.

Щоб ефективно використовувати розподілені дані та спеціалізовані апаратні прискорювачі в контексті наукових обчислень, можливо, варто розглянути використання комбінації інструментів та бібліотек, щоб задовольнити ваші конкретні потреби.

Стандартизація API та підтримка NumPy-подібних інтерфейсів у нових бібліотеках грають важливу роль у сприянні адаптації користувачів та запобіганні фрагментації екосистеми. Стабільні та добре задокументовані інтерфейси, схожі на ті, що має NumPy, роблять перехід між різними бібліотеками менш болісним та знижують бар'єри для користувачів.

Це важливо, оскільки виникнення та припинення розвитку різних бібліотек може призвести до проблеми "застою" в області використання нових технологій. Стандартизація дозволяє спільноті легше робити вибір між різними інструментами та обговорювати їхні переваги та недоліки на основі спільних параметрів.

Проте, важливо також враховувати, що інновації і спроби вдосконалити функціональність можуть привести до розвитку нових інтерфейсів чи бібліотек, і це може вплинути на екосистему. Все ж, баланс між стандартизацією та інновацією є важливим аспектом розвитку області наукових обчислень.

Такий центральний координаційний механізм із чітко визначеним API може значно полегшити роботу з різними об’єктами масиву та робити код більш переносним між різними середовищами. Це також відкриває можливості для використання різних обчислювальних ресурсів, таких як GPU чи розподілені системи, без необхідності значних змін у вихідному коді.

Однак слід зауважити, що поки що NumPy не є ідеальним у роботі з усіма можливими апаратними прискорювачами, такими як GPU чи TPU. Деякі нові бібліотеки та фреймворки, зокрема призначені для машинного навчання, розроблені з урахуванням специфічних можливостей цих апаратних засобів та можуть надавати більш ефективні рішення для конкретних завдань.

Усе ж, поступовий розвиток та розширення можливостей бібліотек, таких як NumPy, у напрямку підтримки різних обчислювальних архітектур, може принести значний виграш у гнучкості та ефективності при обробці даних та виконанні обчислень.

Протоколи у NumPy дійсно створюють можливості для більш гнучкої та ефективної роботи з різними бібліотеками та фреймворками. Це дозволяє розширювати обчислювальні можливості, використовуючи різні апаратні ресурси та розподілені системи, забезпечуючи при цьому зручний та стандартизований інтерфейс для користувача.

Протоколи NumPy дозволяють більш гладким та ефективним способом взаємодіяти з іншими бібліотеками та фреймворками, роблячи код більш переносним та готовим до масштабування. Це особливо важливо у контексті сучасних обчислювальних архітектур, де використання різних апаратних прискорювачів та розподілених систем стає все більш поширеним.

3.2 Реалізація методу автоматизованого керування автотранспорту

Для початку було вирішено почати з розробки самого середовища для навчання моделі методу атоматизованого керування автотранспорту за допомогою штучного інтелекту.

Головним об`єктом в середовищі виступає автомобіль, який навчається навичкам водіння на дорожній смузі. За нагороду для автомобіля було обрано використовувати систему «воріт», які встановлює користувач вручну. Кожний раз, коли автомобіль перетинає ці позначки, він отримує нагороду, тим самим починає розуміти що від нього вимагається. За штраф виставлено бокові лінії дорожньої смуги, яку також встановлює користувач (рисунок 3.1).

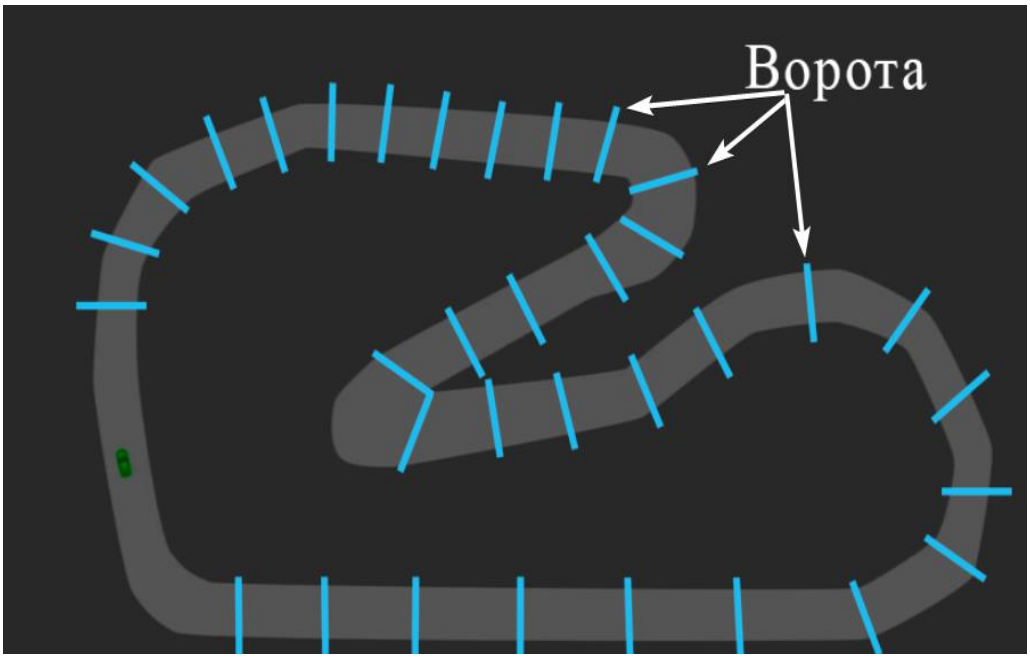


Рис 3.1. Інтерфейс розробленого методу автоматизованого керування автотранспорту за допомогою штучного інтелекту.

Так звані «ворота» та дорожня смугу з її краями виставляються вручну користувачем (рисунок 3.2)

```
def set_walls(self):
    self.walls.append(Wall(240, 809, 200, 583))
    self.walls.append(Wall(200, 583, 218, 395))
    self.walls.append(Wall(218, 395, 303, 255))
    self.walls.append(Wall(303, 255, 548, 173))
    self.walls.append(Wall(548, 173, 764, 179))
    self.walls.append(Wall(764, 179, 1058, 198))
    self.walls.append(Wall(1055, 199, 1180, 215))
    self.walls.append(Wall(1177, 215, 1220, 272))
```

Рис 3.2. Встановлення країв дорожньої смуги

У даному фрагменті коду реалізовано метод, який ініціалізує стіни в середовищі автомобільної моделі. Кожна стіна представлена об'єктом класу Wall і має визначені координати та розміри. Метод додає ці об'єкти до списку стін, задаючи тим самим географічні обмеження та межі для подальшого руху автомобіля в навчальному середовищі.

Для реалізації нагороди для моделі, було створено метод set_gates (рисунок 3.3) , який ініціалізує "ворота винагороди" в середовищі для автомобільної моделі. Кожне ворото представлено об'єктом класу RewardGate з вказаними координатами. Метод додає ці об'єкти до списку воріт, що визначає їх місцезнаходження, через які автомобіль може пройти для отримання винагороди. Ці ворота визначають ключові точки в середовищі, які автомобіль має намір пройти для успішного завершення завдання.

```
def set_gates(self):

    self.gates.append(RewardGate(212, 645, 288, 634))
    self.gates.append(RewardGate(206, 518, 279, 526))
    self.gates.append(RewardGate(224, 390, 286, 416))
    self.gates.append(RewardGate(302, 261, 369, 314))
    self.gates.append(RewardGate(545, 175, 561, 236))
    self.gates.append(RewardGate(846, 182, 841, 259))
    self.gates.append(RewardGate(1114, 203, 1100, 282))
    self.gates.append(RewardGate(1217, 297, 1113, 300))
    self.gates.append(RewardGate(1185, 403, 1102, 339))
    self.gates.append(RewardGate(1042, 462, 979, 408))
```

Рис 3.3. Ініціалізація системи нагород моделі.

Для орієнтації автомобіля в середовищі, а саме пересічення ним краю дорожньої смуги, або отримання нагород, було розроблено наступну математичну модель:

$$A = \frac{\big((x4 - x3) * (y1 - y3) - (y4 - y3) * (x1 - x3)\big)}{\big((y4 - y3) * (x2 - x1) - (x4 - x3) * (y2 - y1)\big)}, \tag{3.1}$$

$$B = \frac{((x2 - x1) * (y1 - y3) - (y2 - y1) * (x1 - x3))}{((y4 - y3) * (x2 - x1) - (x4 - x3) * (y2 - y1))}, \tag{3.2}$$

Якщо $0 \leq A \leq 1$ та $0 \leq B \leq 1$, то точки пересікаються

де A,B - параметри, які використовуються для визначення точки перетину двох відрізків;

x1,y1,x2,y2 – координати кінцевих точок першого відрізку;

x3,y3,x4,y4 – координати кінцевих точок другого відрізку.

Для обчислення відстані між двома точками застосовується наступна формула (3.3):

$$D = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}, \tag{3.3}$$

де D – відстань між двома точками;

x1,y1 – координати першої точки;

x2,y2 – координати другої точки.

Реалізація методу для відсліковування зіткнення автомобіля представлено на рисунку 3.4.

```
def hitCar(self, car):
    global vec2
    cw = car.width
    ch = car.height
    rightVector = vec2(car.direction)
    upVector = vec2(car.direction).rotate(-90)
    carCorners = []
    cornerMultipliers = [[1, 1], [1, -1], [-1, -1], [-1, 1]]
    carPos = vec2(car.x, car.y)
    for i in range(4):
        carCorners.append(carPos + (rightVector * cw / 2 * cornerMultipliers[i][0]) +
                          (upVector * ch / 2 * cornerMultipliers[i][1]))

    for i in range(4):
        j = i + 1
        j = j % 4
        if linesCollided(self.x1, self.y1, self.x2, self.y2, carCorners[i].x, carCorners[i].y, carCorners[j].x,
                         carCorners[j].y):
            return True
    return False
```

Рис 3.4. Реалізація зіткення автомобіля з краєм дорожньої смуги.

У даному фрагменті коду реалізовано метод, який визначає можливі зіткнення між краєм дорожньої смуги та та автомобілем. Для цього використовуються координати та розміри автомобіля, а також вектори, які вказують вправо та вгору відносно його напрямку руху. Метод обчислює координати кутів автомобіля та перевіряє пари сусідніх кутів на можливі зіткнення з лінією. Якщо хоча б одна пара кутів перетинається з лінією, метод повертає True, вказуючи на зіткнення, в іншому випадку повертає False. Цей підхід використовується для визначення подій зіткнень автомобіля з об'єктами в грі та управління поведінкою автомобіля в залежності від зіткнень.

Метод checkRewardGates (рисунок 3.5) в класі автомобіля перевіряє можливе зіткнення з воротами винагороди. Якщо зіткнення виявлено, ворота

вимикаються, збільшується рахунок та призначається винагорода. Якщо гравець пройшов усі ворота, вони реактивуються, а номер воріт скидається.

```
def checkRewardGates(self):
    global vec2
    self.reward = -1
    if self.rewardGates[self.rewardNo].hitCar(self):
        self.rewardGates[self.rewardNo].active = False
        self.rewardNo += 1
        self.score += 1
        self.reward = 10
        if self.rewardNo == len(self.rewardGates):
            self.rewardNo = 0
            for g in self.rewardGates:
                g.active = True
    self.directionToRewardGate = self.rewardGates[self.rewardNo].center - vec2(self.x, self.y)
```

Рис 3.5. Реалізація проходження воріт нагород.

Для формування «мозку» штучного інтелекту автомобіля була створена схема, яка продемонстрована на рисунку 3.6.

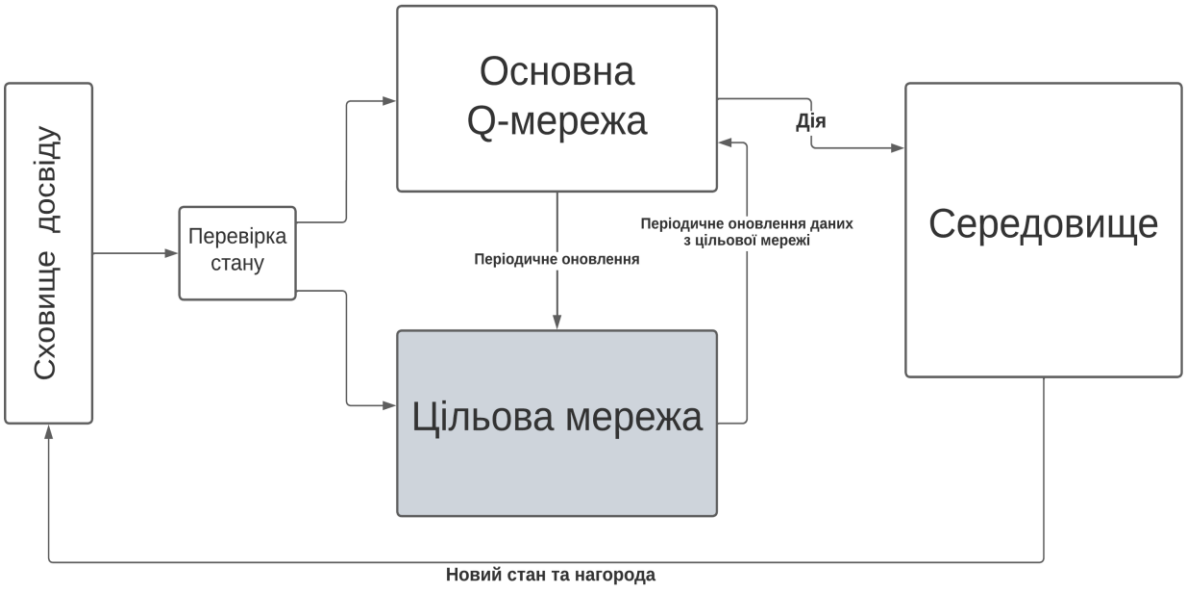


Рис 3.6. Схема роботи методу автоматизованого керування автотранспортом за допомогою штучного інтелекту.

Схема показує, як агент взаємодіє з ігровою середою. Агент приймає дію в поточному стані, яка призводить до нового стану та нагороди. Нагорода представляє собою оцінку того, наскільки добре агент виконав дію.

Агент запам'ятовує досвід взаємодії з ігровою середою в пам'яті досвіду. Цей досвід використовується для навчання агента на основі накопиченого досвіду.

Цільова мережа використовується для стабілізації навчання Q-навчання. Основна мережа постійно оновлюється на основі нового досвіду, який отримує агент. Це може призвести до того, що основна мережа буде швидко змінюватися, що може ускладнити навчання. Цільова мережа допомагає стабілізувати навчання, надаючи більш стабільну мету для оновлення основної мережі.

Цільова мережа оновлюється з меншою частотою, ніж основна мережа. Це дозволяє основній мережі вчитися на основі більш стабільних оцінок Q-значень, які надає цільова мережа.

Ініціалізація агента для керування транспортним засобом представлена на рисунку 3.7.

```
self.sess = tf.compat.v1.Session()

self.DQNetwork = DQN(self.stateSize, self.actionSize, self.learningRate, name='DQNetwork')
self.TargetNetwork = DQN(self.stateSize, self.actionSize, self.learningRate, name='TargetNetwork')

self.memoryBuffer = PrioritisedMemory(self.memorySize)
self.pretrain()

self.state = []
self.trainingStepNo = 0

self.newEpisode = False
self.stepNo = 0
self.episodeNo = 0
self.saver = tf.compat.v1.train.Saver()

load = False
loadFromEpisodeNo = 15800
if load:
    self.episodeNo = loadFromEpisodeNo
    self.saver.restore(self.sess, "./allModels/modelMatin{}/models/model.ckpt".format(self.episodeNo))
else:
    self.sess.run(tf.compat.v1.global_variables_initializer())
# self.sess.graph.finalize()
self.sess.run(self.update_target_graph())
```

Рис 3.7. Створення агента для навчання з підкріпленням.

У цьому фрагменті коду створюється агент для навчання з підкріпленням на основі Deep Q-Network (DQN). Перш за все, визначається TensorFlow сесія, після чого ініціалізуються основна та цільова мережі DQN. Буфер пам'яті створюється для зберігання даних про попередні дії агента. Метод pretrain() викликається для попереднього навчання та заповнення буфера пам'яті початковими даними. Змінні, такі як self.state, self.trainingStepNo, self.newEpisode, self.stepNo, та self.episodeNo, слідкують за поточним станом та ітераціями навчання. Об'єкт self.saver використовується для зберігання та відновлення моделі. Вибір між новим навчанням та завантаженням попередньо навченої моделі здійснюється за допомогою умови if load:. На завершення, глобальні змінні TensorFlow ініціалізуються, а також викликається метод update_target_graph() для оновлення цільової мережі з метою забезпечення збіжності DQN.

Метод update_target_graph (рисунок 3.8), в класі агента виконує оновлення параметрів цільової мережі Deep Q-Network (DQN) з основної мережі. Спочатку визначаються ваги та зміщення обох мереж, і для кожного відповідного параметра цільової мережі створюється операція, що призначає йому значення від основної мережі. Це забезпечує періодичне оновлення параметрів цільової мережі, що

сприяє стабільнішому та ефективнішому навчанню агента в середовищі з підкріпленням. Такий механізм оновлення є ключовим елементом розробленого методу, допомагаючи агентові адаптуватися до змінюючогося середовища та покращувати його стратегію взаємодії з оточенням.

```python
def update_target_graph(self):

    # Отримання параметрів з основної мережі
    from_vars = tf.compat.v1.get_collection(tf.compat.v1.GraphKeys.TRAINABLE_VARIABLES, "DQNetwork")
    # Отримання параметрів з цільової мережі
    to_vars = tf.compat.v1.get_collection(tf.compat.v1.GraphKeys.TRAINABLE_VARIABLES, "TargetNetwork")

    op_holder = []

    # Оновлення даних цільової мережі з основної
    for from_var, to_var in zip(from_vars, to_vars):
        op_holder.append(to_var.assign(from_var))
    return op_holder
```

Рис 3.8. Оновлення значень цільової мережі з основної мережі.

Оновлення цільової мережі з основної мережі в алгоритмі DQN допомагає зробити навчання агента більш стійким і ефективним. Воно зменшує коливання та ризик великих помилок, що можуть виникнути під час навчання. Оновлення цільової мережі дає агентові більш стабільні цільові значення для використання під час навчання, допомагаючи йому швидше знаходити оптимальні стратегії в середовищі. Це сприяє стабільності та прискорює процес досягнення мети в задачах навчання з підкріпленням.

```
def train(self):

    if self.trainingStepNo == 0:
        self.state = self.game.get_state()

    if self.newEpisode:
        self.state = self.game.get_state()

    if self.stepNo < self.maxSteps:
        self.stepNo += 1
        self.decayStep += 1
        self.trainingStepNo += 1
        self.tau += 1

        epsilon = self.minEpsilon + (self.maxEpsilon - self.minEpsilon) * np.exp(
            -self.decayRate * self.decayStep)

        if np.random.rand() < epsilon:
            choice = random.randint(1, len(self.possibleActions)) - 1
            action = self.possibleActions[choice]

        else:
            QValues = self.sess.run(self.DQNetwork.output,
                                    feed_dict={self.DQNetwork.inputs_: np.array([self.state])})
            choice = np.argmax(QValues)
            action = self.possibleActions[choice]

        actionNo = np.argmax(action)
        reward = self.game.make_action(actionNo)
```

Рис 3.9. Початкове навчання агента

Метод train визначає процес навчання агента (рисунок 3.9). По-перше, відбувається ініціалізація стану гри, що є необхідним на початку навчання та після завершення кожного епізоду. Після цього визначається експлораційна ймовірність для випадкового вибору дії, яка зменшується з часом. Залежно від цієї ймовірності агент вибирає або випадкову дію, або оптимальну дію, основану на поточній стратегії. Обрана дія виконується в грі, і агент отримує відповідну винагороду. Цей процес допомагає агентові взаємодіяти з середовищем, вдосконалювати свою стратегію та максимізувати здобуту винагороду під час навчання в грі.

```
treeIndexes, batch, ISWeights = self.memoryBuffer.sample(self.batchSize)

statesFromBatch = np.array([exp[0][0] for exp in batch])
actionsFromBatch = np.array([exp[0][1] for exp in batch])
rewardsFromBatch = np.array([exp[0][2] for exp in batch])
nextStatesFromBatch = np.array([exp[0][3] for exp in batch])
carDieBooleansFromBatch = np.array([exp[0][4] for exp in batch])

targetQsFromBatch = []

QValueOfNextStates = self.sess.run(self.TargetNetwork.output,
                                   feed_dict={self.TargetNetwork.inputs_: nextStatesFromBatch})

for i in range(self.batchSize):
    action = np.argmax(QValueOfNextStates[i])
    terminalState = carDieBooleansFromBatch[i]
    if terminalState:
        targetQsFromBatch.append(rewardsFromBatch[i])
    else:
        target = rewardsFromBatch[i] + self.gamma * QValueOfNextStates[i][action]
        targetQsFromBatch.append(target)

targetsForBatch = np.array([t for t in targetQsFromBatch])
```

Рис 3.10. Підготовка до оптимізації Q-мережі

У даному фрагменті коду проводиться підготовка даних для оптимізації Q-мережі на основі вибраного пакету досвіду. Спершу вибирається пакет досвіду з пам'яті агента, з якого розпаковуються окремі компоненти, такі як стани, дії, винагороди, наступні стани та інформація про завершення епізоду.

Далі використовується цільова мережа для обчислення Q-значень наступних станів. Для кожного елементу пакету визначається оптимальна дія та обчислюється цільове Q-значення. Якщо наступний стан є термінальним , а саме відповідає завершенню епізоду, цільове Q-значення встановлюється рівним винагороді.

Отримані цільові Q-значення формують масив, який використовується у процесі оптимізації параметрів Q-мережі. Цей етап є ключовим для покращення точності та ефективності навчання агента для керування автотранспортом, відповідно до отриманого досвіду.

```
    self.memoryBuffer.batchUpdate(treeIndexes, absoluteErrors)

if self.stepNo >= self.maxSteps:
    self.episodeNo += 1
    self.stepNo = 0
    self.newEpisode = True
    self.game.new_episode()
    if self.episodeNo >= self.totalTrainingEpisodes:
        self.training = False
    if self.episodeNo % 100 == 0:
        directory = "./allModels/model{}".format(self.episodeNo)
        if not os.path.exists(directory):
            os.makedirs(directory)
        save_path = self.saver.save(self.sess,
                                    "./allModels/model{}/models/model.ckpt".format(self.episodeNo))
        print("Model Saved")
if self.tau > self.maxTau:
    self.sess.run(self.update_target_graph())
    self.tau = 0
    print("Target Network Updated")
```

Рис 3.11. Оновлення пам`яті та керування епохами.

Деякі важливі аспекти роботи методу автоматизованого керування автомобілем за допомогою штучного інтелекту представлено на рисунтку 3.11. Перш за все, виконується оновлення пам'яті з пріоритетами, де ваги досвіду коригуються на основі абсолютних помилок. Далі відбувається контроль кількості кроків та початок нової епохи, що визначається, чи не перевищено кількість кроків у поточній серії. Якщо це так, ініціюється новий епізод в середовищі. Після цього перевіряється, чи досягнута загальна кількість епох для завершення навчання. Модель також зберігається кожні 100 епох для подальшого використання. Крім того, періодично оновлюється цільова мережа для підтримки стабільності навчання. У кінці, код включає логіку для оновлення параметрів, зупинки навчання і збереження моделі за визначеними умовами.

3.3 Аналіз результатів

Для аналізу результатів дослідження, а саме виявлення мінімального відхилення системи керування автотратранспортом за допомогою штучного

інтелекту, було вирішено провести аналіз 3х агентів. Перший агент використовує класичний метод Deep Q-Network (DQN), який є відомим та широко застосовуваним в галузі машинного навчання. Другий агент використовує створений метод, розроблений на базі Q-навчання , що включає в себе вдосконалений алгоритм роботи. Третій агент буде визначений як контрольний, де керування автомобілем здійснюється людиною.

Аналіз буде проведено за допомогою графіків. Графік представляє собою «дорожню смугу», де вся його довжина, є маршрутом транспортного засобу. Значення 1 та -1 виступають боковими лініями дорожньої смуги, пересічення яких вважається виїздом за її межу. Значенням 0 виступає центр заданого маршруту для транспортного засобу.

Першим було проведено аналіз класичного методу Deep Q Network, що представлено на рисунку 3.12.

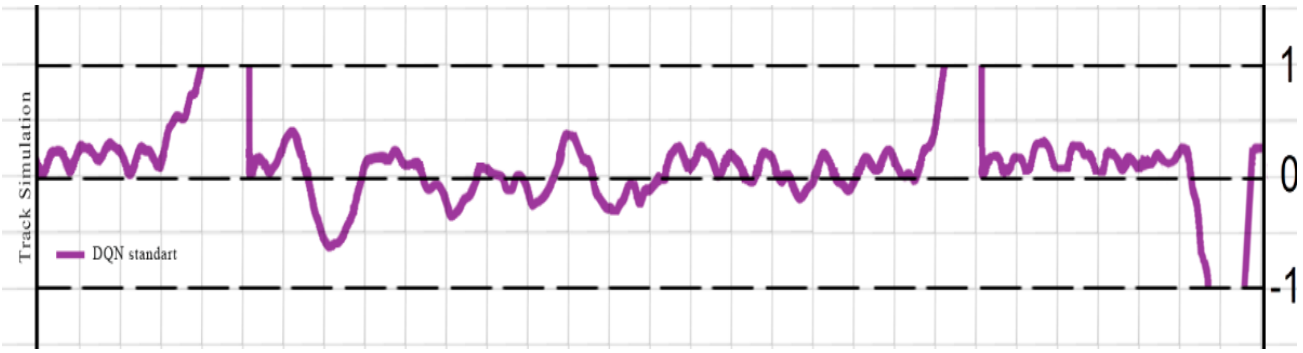


Рис 3.12. Симуляція траєкторії Deep Q Network

На графіку ми можемо спостерігати, що через основний недолік методу Deep Q Network, а саме відсутніх цільових значень, через їх постійну і безперервну зміну, автомобіль час від часу вилітає за дорожню смугу, особливо цю проблему можна спостерігати на поворотах заданого маршруту.

Наступним було проведено симуляцію траєкторії за допомогою створеного методу автоматизованого керування автотранспотром, що представлено на рисунку 3.13.

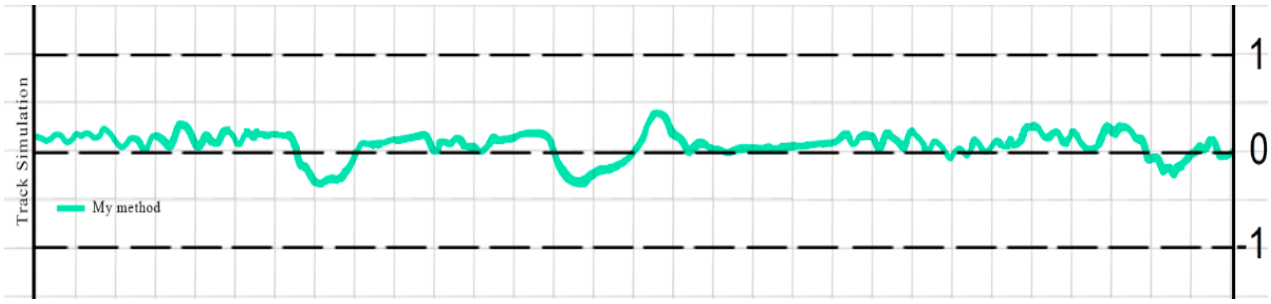


Рис 3.13. Симуляція траєкторії розробленого методу

На представленому графіку можна візуально спостерігати, що створений метод справляється краще, ніж стандартний метод Deep Q Network, та намагається дотримуватись центру заданого користувачем маршруту.

З контрольною симуляцією траекторії руху, а саме коли маршрут проїзжає людина можна ознайомитись на рисунку 3.14.

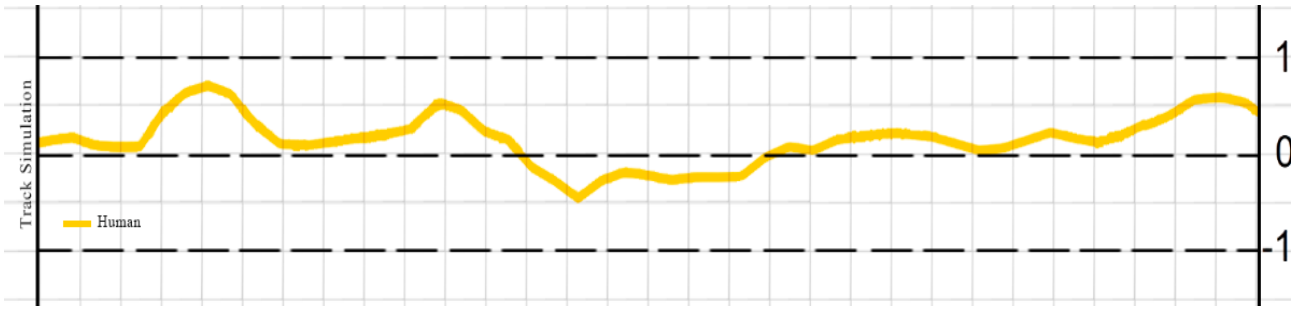


Рис 3.14. Траєкторія руху при керуванні людиною

При поєднанні всіх 3х графіків ми отримуємо такий результат (рисунок 3.15). Для виявлення мінімального відхилення від заданої траекторії системи керування автотранспортом за допомогою штучного інтелекту, було вирішено прорахувати середнє відхилення кожної симуляції протягом всього маршруту.

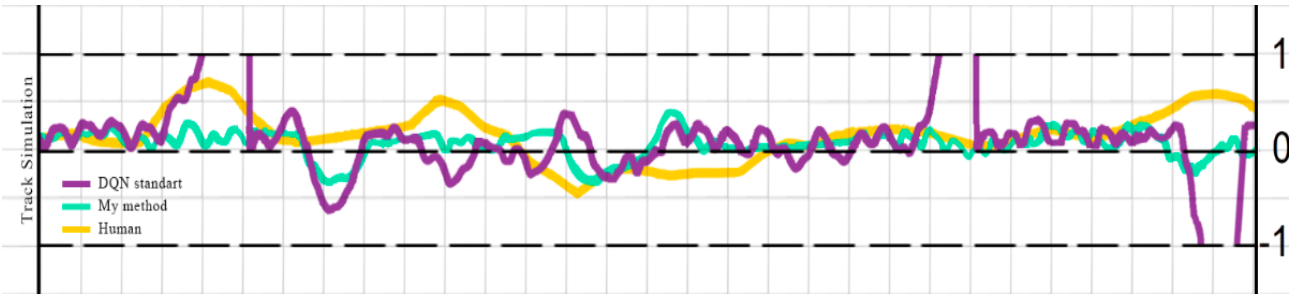


Рис 3.15. Результат симуляції руху автотранспорту на заданому маршруті.

Таблиця 3.1. Середнє відхилення авто від заданого маршруту

Метод	Середнє відхилення в м.	Середнє відхилення в %
Deep Q Network	0.56	56
My method	0.17	17
Human	0.22	22

Після аналізу результатів дослідження було виявлено, що створений метод має мінімальне відхилення від заданої траєкторії під час автоматизованого керування автотранспортом за допомогою штучного інтелекту, а саме середнє відхилення в 0.17 метрів, що виступає найкращим результатом серед досліджених методів.

ВИСНОВКИ

В результаті виконання магістерської роботи було розроблено метод автоматизованого керування автотранспортом за допомогою штучного інтелекту на прикладі відеогри, основною метою якого було, мінімізація відхилення від заданої траєкторії під час автоматизованого керування автотранспортом.

При виконанні роботи було виконано наступні задачі.

1. Проаналізовано існуючі підходи до побудови та використання штучного інтелекту в автоматизованих системах керування автотранспортом, зокрема, в процесі їх переміщення та орієнтуванні в просторі під час дорожнього руху.

2. Обрано за основу тип навчання з підкріпленням, а саме метод Q-навчання, на основі якого розроблено модель поведінки автотранспорту на заданій траєкторії.

3. Розроблено модель поведінки автотранспорту на заданому маршруті, що враховує за винагороду виставлені користувачем «ворота» нагороди, та виїзд з проїзної частини за штраф.

4. Розроблено та проведено тестування програмного забезпечення, що моделює поведінку автотранспорту на заданому маршруті, аналізує та зберігає інформацію проміжних даних та результатів.

5. Проведено моделювання систем автоматизованого керування автотранспортом на заданому маршруті для проведення аналізу отриманих результатів. За результатами аналізу виявлено, що розроблений метод має мінімальне відхилення 0.17 метрів від заданого маршруту, у порівнянні з іншими моделями, що були проаналізовані.

ПЕРЕЛІК ПОСИЛАНЬ

1. Alpaydın E. Machine learning. Wiley Interdisciplinary Reviews: Computational Statistics. 2011. - С. 195–203.
2. Supervised learning / D. Valkenborg та ін. American Journal of Orthodontics and Dentofacial Orthopedics. 2023. - С. 146–149.
3. Google Cloud What is unsupervised learning? [Електронний ресурс]: – Режим доступу: https://cloud.google.com/discover/what-is-unsupervised-learning 15.10.2023
4. Nandy A., Biswas M. RL Theory and Algorithms. Reinforcement Learning. Berkeley, 2017. - С. 19–69.
5. McCosker A., Wilken R. How Does a Car Learn to See?. Automating Vision. 2020.- С. 94–111.
6. Чень, С., Цзянь, З., Хуан, Ю., Чень, Ю., Чжоу, З. та Чжен, Н. Автономне водіння: когнітивне конструювання та розуміння ситуації. 2019. – С.82-85.
7. Review on Lidar Technology [Електронний ресурс]: – Режим доступу: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3604309 16.10.2023
8. The Basics of LiDAR - Light Detection and Ranging [Електронний ресурс]: – Режим доступу: https://www.neonscience.org/resources/learning-hub/tutorials/lidar-basics 16.10.2023
9. MEMS-based lidar for autonomous driving [Електронний ресурс]: – Режим доступу: https://link.springer.com/article/10.1007/s00502-018-0635-2 17.10.2023
10. Xu G., Xu Y. Applications of GPS Theory and Algorithms. GPS. Berlin, Heidelberg, 2016 - С. 313–340.
11. Андерсон, Дж. М., Калра, Н., Стенлі, К. Д., Соренсен, П., Олуватола, О. А. та Корпорація Ренд. Технологія автономних транспортних засобів: посібник для політиків 2016.- С.45-56
12. Self-driving car [Електронний ресурс]: – Режим доступу:

https://www.techtarget.com/searchenterpriseai/definition/driverless-car 17.10.2023

13. Jacobson L. GNSS Markets and Applications (GNSS Technology and Applications). Artech House Publishers, 2007 - 216 с.
14. AbuSalim, SWG, Ibrahim, R., Saringat, MZ, Jamel, S. and Wahab, JA. Порівняльний аналіз між алгоритмами Дейкстри та Беллмана-Форда в оптимізації найкоротшого шляху, 2020 – С.53-67.
15. Camacho E. F., Bordons C. Introduction to Model Based Predictive Control. Model Predictive Control. London, 2018. С. 34–45.
16. How cities can benefit from automated driving [Електронний ресурс]: – Режим доступу: https://www.bosch.com/stories/economic-impact-of-self-driving-cars/ 17.10.2023
17. Greenhouse Gas Emissions from a Typical Passenger Vehicle [Електронний ресурс]: – Режим доступу: https://www.epa.gov/greenvehicles/greenhouse-gas-emissions-typical-passenger-vehicle 18.10.2023
18. Francisco S. Melo Convergence of Q-learning: a simple proof. Lisboa, Portugal, 2019. С. 2-4.
19. Ajay Kumar. Introduction to Hybrid algorithms. International Journal of Advanced Research in Science, Communication and Technology. 2022. С. 604–610.
20. Hackbusch W. Gradient Method. Applied Mathematical Sciences. Cham, 2016. С. 211–228.
21. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. 2013. С. 1-7.
22. Milosevic N. Exposing Neural Network Models. Introduction to Convolutional Neural Networks. Berkeley, CA, 2020. С. 122-125
23. Lample G., Chaplot D. S. Playing FPS Games with Deep Reinforcement Learning. Proceedings of the AAAI Conference on Artificial Intelligence. 2017. С. 52-61

24. InSight Mission Overview [Електронний ресурс]: – Режим доступу: https://mars.nasa.gov/insight/mission/overview/ 18.10.2023
25. Huang Y. Deep Q-Networks. Deep Reinforcement Learning. Singapore, 2020. С. 135–160
26. Sewak M. Deep Q Network (DQN), Double DQN, and Dueling DQN. Deep Reinforcement Learning. Singapore, 2019. С. 95–108
27. Target Extraction. Encyclopedia of Social Network Analysis and Mining. New York, NY, 2018. С. 3039
28. Naylor M. Decision Tree. Mathematics Teacher: Learning and Teaching PK-12. 2020. С. 113
29. Tensorflow About [Електронний ресурс]: – Режим доступу: https://www.tensorflow.org/about 18.10.2023
30. Tensorflow Learn [Електронний ресурс]: – Режим доступу: https://www.tensorflow.org/learn 19.10.2023
31. Scarpino M. TensorFlow. Hoboken, New Jersey : John Wiley & Sons, Inc., 2018 – С. 335-336.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ
(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«Розробка методу автоматизованого керування автотранспортом за допомогою штучного інтелекту на прикладі відеогри»

Виконав: студент групи ПДМ-64 Гаврилюк Валерій Олександрович

Керівник: к.т.н., доц., доцент кафедри Трінтіна Наталія Альбертівна

Київ - 2024

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: мінімізація відхилення від заданої траекторії в процесі автоматизованого керування автотранспортом.

Об'єкт дослідження: процес автоматизованого керування автотранспортом.

Предмет дослідження: Методи автоматизованого керування автотранспортом за допомогою штучного інтелекту.

ПОРІВНЯННЯ ПАРАДИГМ МАШИННОГО НАВЧАННЯ

НАЗВА	ПЕРЕВАГИ	НЕДОЛІКИ
Навчання з учителем	Висока точність	Необхідність наявності набору позначених даних
	Швидке навчання	Чутливість до шуму в даних
	Можливість інтерпретації результатів	
Навчання без учителя	Не потрібен позначений набір даних	Має меншу точність навчання
	Широкий спектр застосування	Складність інтерпретації результатів
	Пошук прихованих шаблонів	
Навчання з підкріпленням	Адаптивність в складних середовищах	Нестабільність навчання
	Навчання на основі власних даних	Вибагливість до ресурсів
	Ефективність обробки даних про стани середовища	

3

МАТЕМАТИЧНА МОДЕЛЬ ОРІЄНТУВАННЯ АВТОТРАНСПОРТУ В СЕРЕДОВИЩІ

$$A = \frac{((x4 - x3) * (y1 - y3) - (y4 - y3) * (x1 - x3))}{((y4 - y3) * (x2 - x1) - (x4 - x3) * (y2 - y1))},$$

$$B = \frac{((x2 - x1) * (y1 - y3) - (y2 - y1) * (x1 - x3))}{((y4 - y3) * (x2 - x1) - (x4 - x3) * (y2 - y1))},$$

Якщо $0 \le A \le 1$ та $0 \le B \le 1$, то точки пересікаються

де A,B - параметри, які використовуються для визначення точки перетину двох відрізків;

x1,y1,x2,y2 – координати кінцевих точок першого відрізку;

x3,y3,x4,y4 – координати кінцевих точок другого відрізку.

$$D = \sqrt{(x2 - x1)^2 + (y2 - y1)^2},$$

де D – відстань між двома точками;

x1,y1 – координати першої точки;

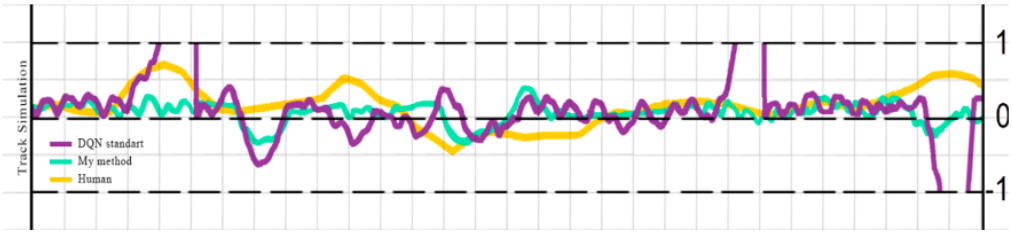
x2,y2 – координати другої точки.

4

СХЕМА МЕТОДУ АВТОМАТИЗОВАНОГО КЕРУВАННЯ АВТОТРАНСПОРТУ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

5

РЕЗУЛЬТАТ ПОРІВНЯННЯ МЕТОДІВ



Метод	Середнє відхилення в м.	Середнє відхилення в %
Deep Q Network	0.56	56
My method	0.17	17
Human	0.22	22

6

ПРАКТИЧНИЙ РЕЗУЛЬТАТ

7

ВИСНОВКИ

1. Проаналізовано існуючі підходи до побудови та використання штучного інтелекту в автоматизованих системах керування автотранспортом, зокрема, в процесі їх переміщення та орієнтуванні в просторі під час дорожнього руху.
2. Обрано за основу тип навчання з підкріпленням, а саме метод Q-навчання, на основі якого розроблено модель поведінки автотранспорту на заданій траєкторії.
3. Розроблено модель поведінки автотранспорту на заданому маршруті, що враховує за винагороду виставлені користувачем «ворота» нагороди, та виїзд з проїзної частини за штраф.
4. Розроблено та проведено тестування програмного забезпечення, що моделює поведінку автотранспорту на заданому маршруті, аналізує та зберігає інформацію проміжних даних та результатів.
5. Проведено моделювання систем автоматизованого керування автотранспортом на заданому маршруті для проведення аналізу отриманих результатів. За результатами аналізу виявлено, що розроблений метод має мінімальне відхилення 0.17 метрів від заданого маршруту, у порівнянні з іншими моделями, що були проаналізовані.

8

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Статті:

1. Гаврилюк В.О., Трінтіна Н.А. Застосування штучного інтелекту в системах управління транспортними засобами // Зв'язок. Подано до друку.

Тези доповідей:

1. Гаврилюк В.О., Дослідження ролі технології штучного інтелекту в автомобільній індустрії / V Всеукраїнська конференція молодих учених «Актуальні питання розвитку інформаційних технологій» Збірник тез 22.11.2023, ПДТУ, м. Дніпро – К.: ПДТУ, 2023. С. 61-62.
2. Гаврилюк В.О., Дослідження ключових аспектів розвитку машинного навчання / V Всеукраїнська конференція молодих учених «Актуальні питання розвитку інформаційних технологій» Збірник тез 22.11.2023, ПДТУ, м. Дніпро – К.: ПДТУ, 2023. С.63-64.