СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування

Пояснювальна записка

до магістерської дипломної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему Технологія впровадження штучного інтелекту в автоматизоване пілотування транспортним засобом

Виконав: студент 2 курсу, групи ІСТ-22дм
126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

Рагулін М. В.
(прізвище та ініціали)

Керівник Лифар В. О.
(прізвище та ініціали)

Рецензент Меняйленко О.С.
(прізвище та ініціали)

Київ – 2023 року

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ ДО МАГІСТЕРСЬКОЇ ДИПЛОМНОЇ РОБОТИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

Освітньо-кваліфікаційний рівень магістр

Спеціальність 126 «Інформаційні системи та технології»

(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ д.т.н., доц. Лифар В.О.
(підпис)
«____»_____202__р.

ЗАВДАННЯ
на магістерську дипломну роботу студенту
Рагулін Максим Вікторович
(прізвище, ім'я, по батькові)

1. Тема роботи Технологія впровадження штучного інтелекту в автоматизоване пілотування транспортним засобом
керівник роботи Лифар Володимир Олексійович, д.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «___»_____202_року №____

2. Строк подання студентом роботи 06.12.2023

3. Вихідні дані до роботи_____

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

5. Перелік графічного матеріалу (з точним значенням обов'язків креслень)_____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 18 жовтня 2023

КАЛЕНДАРНИЙ ПЛАН

№ з\п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	23.10.23 – 29.10.23	
2	Пошук та аналіз існуючих рішень	30.10.23 – 02.11.23	
3	Аналіз методів та алгоритмів	02.11.23 – 10.11.23	
4	Розробка архітектури алгоритму	11.11.23 – 15.11.23	
5	Реалізація алгоритму	16.11.23 – 28.11.23	
6	Апробація технології	29.11.23 – 01.12.23	
7	Оформлення пояснювальної записки	02.12.23 – 05.12.23	
8	Підготовка та подання магістерської роботи до захисту	06.12.23 – 06.12.23	

Студент_____ Рагулін М. В.
(підпис) (прізвище та ініціали)

Керівник роботи_____ Лифар В.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дипломна робота: 63 стор., 14 рис., 3 граф., додаток А (основний код), додаток Б (файл налаштувань) 15 джерел.

Об’єкт дослідження – система автоматизованого пілотування для автомобілів з звикористанням штучного інтелекту для оптимізації та управління поведінкою автотранспортних засобів.

Мета роботи – розробка та реалізація системи, яка здатна навчати автомобілі за допомогою алгоритму та забезпечити їх автономний рух, здатний адаптуватися до різних дорожніх ситуацій.

Проведено аналіз стану використання та розвитку штучного інтелекту в автомобільній промисловості. Розглянуто ключові аспекти впровадження ШІ в системи автомобілів та визначено основні можливості цього напряму. Реалізовано систему автоматизованого навчання автомобілів автономного руху з здатністю адаптуватися до різних дорожніх ситуацій та проведено експерименти у стимуляційному середовищі.

НЕЙРОМЕРЕЖА, ШТУЧНИЙ ІНТЕЛЕКТ, АЛГОРИТМ, СИСТЕМА, АВТОПІЛОТ

ЗМІСТ

ВСТУП.....7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....9
1.1 Штучний інтелект (AI).....9
1.1.1 Машинне навчання (ML).....10
1.1.2 Глибоке навчання (DL).....10
1.1.3 Навчання з підкріпленням (RL).....11
1.2 Концепція ШІ.....12
1.2.1 Основні компоненти нейронних мереж:.....13
1.2.2 Типи Нейронних Мереж.....14
РОЗДІЛ 2. ТЕОРЕТИЧНІ АСПЕКТИ ШТУЧНОГО ІНТЕЛЕКТУ В АВТОТРАНСПОРТІ.....16
2.1 Огляд існуючих рішень.....17
2.2 Етичні та юридичні аспекти автономних систем у транспорті.....28
2.3 Допоміжні системи штучного інтелекту орієнтування на дорозі.....30
РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЗОВАНОГО ПІЛОТУВАННЯ.....38
3.1 Середовище розробки та допоміжні бібліотеки.....38
3.1.1 Python.....38
3.1.2 PyCharm.....38
3.1.3 NEAT.....39
3.1.4 Pygame.....39
3.2 Архітектува системи.....39
3.2.1 Основні характеристики та концепції NEAT.....40
3.3 Алгоритм системи.....44
РОЗДІЛ 4. АПРОБАЦІЯ В СИМУЛЯЦІЙНОМУ СЕРЕДОВИЩІ.....48
4.1 Методика апробації.....48
4.2 Оцінка результатів.....50
4.3 Результати апробації.....51
4.4 Перспективи та подальші дослідження.....52
ВИСНОВОК.....53

СПИСОК ЛІТЕРАТУРИ 54
ДОДАТОК А 55
ДОДАТОК Б 62

ВСТУП

Швидкі та вражаючі темпи розвитку технологій в останні десятиліття неодмінно змінюють транспортну індустрію, відкриваючи нові перспективи для реалізації автономних систем управління. Одним із ключових компонентів цього трансформаційного процесу є впровадження штучного інтелекту в автоматизоване пілотування транспортних засобів, що визначає необхідність глибокого аналізу та розробки нових підходів до реалізації цієї технології.

За останні кілька років спостерігається стрімкий розвиток систем штучного інтелекту, який відкриває безліч можливостей для вдосконалення транспортного сектору. Використання штучного інтелекту у сфері автономного транспорту приводить до покращення безпеки, зниження витрат, та оптимізації маршрутів, що робить цю технологію невід'ємною частиною майбутнього транспортної системи.

Основною метою магістерської дипломної роботи є аналіз та розробка системи штучного інтелекту, спрямованої на автоматизоване пілотування транспортного засобу, з фокусом на знаходженні оптимального маршруту. Дослідження передбачає обгрунтування вибору методів машинного навчання та алгоритмів, вивчення сучасних сенсорів, а також апробацію системи в симуляційному середовищі.

Проведення даного дослідження є актуальним не лише в контексті наукового прогресу, але й має значущість для індустріальних галузей, спрямованих на розвиток автономного транспорту. Новизна роботи полягає у поєднанні теоретичних аспектів штучного інтелекту з практичними рішеннями для створення надійної та оптимальної системи управління транспортними засобами.

Дипломна робота складається з чотирьох основних розділів. Розділ 1 розглядає аналіз предметної області та концепцію роботи штучного інтелекту. Розділ 2 розглядає теоретичні аспекти штучного інтелекту в автотранспорті, включаючи огляд існуючих систем автоматизованого пілотування та основні принципи штучного інтелекту Розділ 3 детально описує розробку та реалізацію системи штучного інтелекту для автоматизованого пілотування, включаючи

архітектуру системи, навчання в симуляційному середовищі та інтеграцію сенсорів. Розділ 4 присвячений проведенню експериментів та аналізу отриманих результатів.

Заключні висновки, включають у себе основні висновки з дослідження та підведення підсумків результатів роботи.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Штучний інтелект (AI)

Штучний інтелект (AI) [1] – це галузь інформатики, яка створює та вивчає системи, здатні до виконання завдань, які зазвичай вимагають інтелекту людини. Він включає в себе розробку програм та алгоритмів, які дають комп'ютерам здатність розв'язувати завдання, вимагаючи аналізу, усвідомлення та навчання.

Штучний інтелект спирається на різні підходи та техніки, і одними з ключових галузей є машинне навчання, глибоке навчання та навчання з підкріпленням.

Штучний інтелект[1] знаходить своє застосування в багатьох галузях, включаючи автономні системи, медицину, фінансовий аналіз, розпізнавання мови та зображень, робототехніку та ігрову індустрію. Його ціль - розробка систем, які будуть здатні до виконання завдань швидкіше та точніше, ніж люди, та які будуть здатні навчатися на основі досвіду та покращувати свою продуктивність з часом.

Застосування штучного інтелекту в автотранспорті дозволяє реалізувати автоматизовані системи пілотування, які забезпечують безпеку та ефективність на дорогах. Використання машинного та глибокого навчання дозволяє автоматично адаптуватися до різних умов дорожнього руху та навчатися на прикладах, що покращує якість та надійність системи. Навчання з підкріпленням дозволяє враховувати результати та використовувати систему нагород та покарання для досягнення оптимальних поведінкових моделей.

Розвиток штучного інтелекту забезпечує безліч нових можливостей і викликів[1]. Він перетворює розуміння процесів взаємодії між людьми та комп'ютерами та дає змогу досягнути нових вершин в автоматизації та оптимізації різних галузей. Однак, разом з цим виникають і юридичні, етичні та соціальні питання, які потребують уважного вивчення та вирішення. Штучний інтелект потребує етичного та відповідального підходу використання, що б забезпечити безпеку, прозорість та справедливість його застосування у суспільстві.

1.1.1 Машинне навчання (ML)

Машинне навчання (ML) [7] представляє собою інтелектуальну систему, де комп'ютери вчаться вирішувати завдання, адаптуючи свою діяльність до накопичених даних. Ця галузь штучного інтелекту не просто визначає нові можливості технологій, але також змінює взаємодію людини з технікою.

Однією з ключових концепцій машинного навчання є здатність систем до самостійного навчання без явного програмування. Моделі і алгоритми ML спроможні визначати закономірності в даних, а також вдосконалювати свою продуктивність з кожним новим набором інформації.

У цьому процесі велику роль відіграє нейронна мережа – структура, інспірована роботою людського мозку. Штучні нейрони, об'єднані в мережу, виконують завдання розпізнавання образів, класифікації даних, та навіть вирішення складних завдань, які раніше вважалися неможливими для автоматизації.

1.1.2 Глибоке навчання (DL)

Глибоке навчанн[4] – це важлива галузь машинного навчання, яка використовує глибокі нейронні мережі з числом шарів для аналізу та розуміння складних залежностей в даних. Такий підхід дозволяє побудувати моделі, здатні до автоматичного виявлення та інтерпретації навіть найскладніших патернів.

Основна перевага глибокого навчання полягає в його здатності до «довготривалого запам'ятовування» та «глибокого розуміння контексту». Більш глибокі моделі можуть аналізувати та використовувати більш широкий спектр інформації, що дозволяє їм забезпечувати більш точні результати й робити більш обґрунтовані висновки.

Глибоке навчання[4] застосовується в різних областях машинного навчання, включаючи обробку мови, комп'ютерне зору та автоматичне вирішення завдань, які вимагають глибокого розуміння контексту. Наприклад, в обробці мови глибоке навчання може бути використане для машинного перекладу текстів, розпізнавання

мови та створення голосових помічників. У комп'ютерному зорі глибокі моделі можуть бути використані для розпізнавання об'єктів, виявлення емоцій на обличчі людини та інших завдань комп'ютерного зору.

Області застосування глибокого навчання постійно розширюються, і цей підхід вже має дуже суттєвий вплив на різні галузі, включаючи медицину, фінанси, автомобільну промисловість та багато інших. Його потенціал ще не до кінця досліджений, і з ним пов'язано багато можливостей, які можуть революціонізувати спосіб функціонування та розвитку багатьох галузей.

1.1.3 Навчання з підкріпленням (RL)

Навчання з підкріпленням[2] – інший ключовий підхід, який використовує винагороди та покарання для навчання моделі досягати мети. Цей метод знаходить своє застосування в іграх, робототехніці та стратегічному плануванні.

Машинне навчання розвиває багато галузей, від медицини та фінансів до транспорту та виробництва. Від його успішного впровадження залежить величезний потенціал для автоматизації та оптимізації процесів, що принесе користь суспільству та прискорить науковий прогрес.

Глибоке навчання та навчання з підкріпленням являють собою два важливі підходи у машинному навчанні. Глибоке навчання дозволяє моделям розпізнавати складні залежності та робити обґрунтовані висновки, використовуючи глибокі нейронні мережі. Навчання з підкріпленням використовує винагороди та покарання для навчання моделі досягати поставленої мети. Ці підходи можуть застосовуватись у багатьох галузях, включаючи ігри, робототехніку, медицину та багато інших.

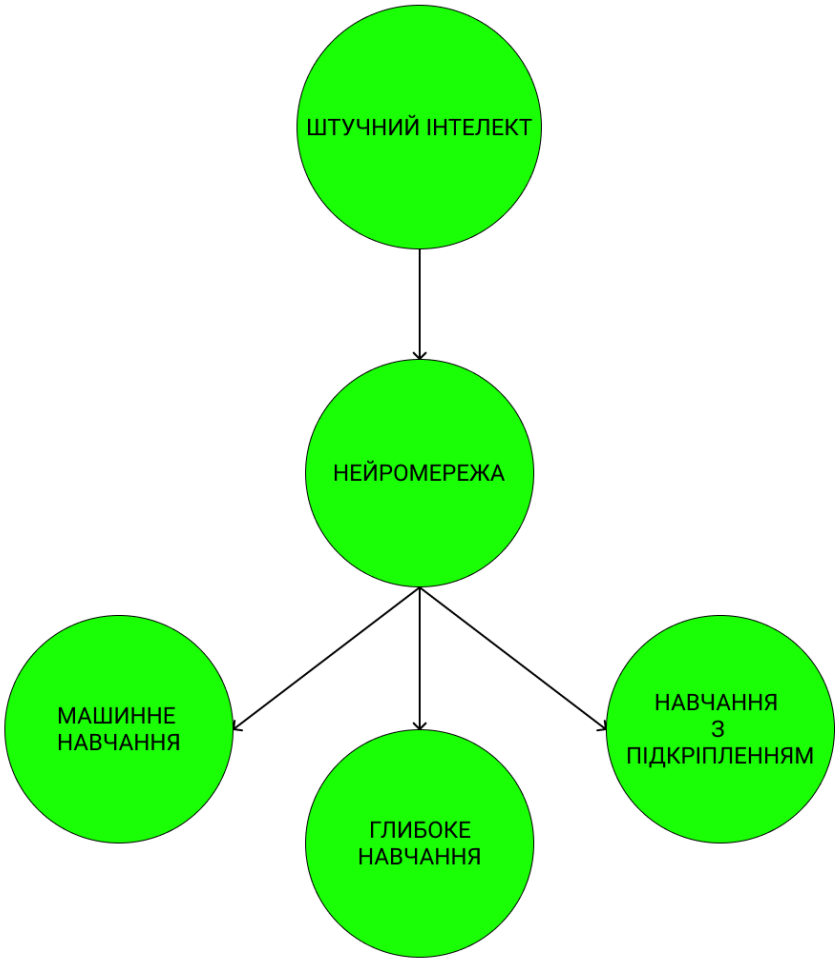


Рис. 1— Складові штучного інтелекту

1.2 Концепція ШІ

Однією з основних концепцій в ШІ є нейронні мережі, які моделюють роботу людського мозку[3]. Нейронні мережі використовуються для розв'язання завдань розпізнавання образів, обробки мови, прогнозування та інших. Вони складаються з великої кількості штучних нейронів, які об'єднані в шари та взаємодіють між собою.

Нейронні мережі є прикладом того, як технологія може імітувати складні процеси людського мозку та вирішувати завдання, які раніше були невиправдано складні для комп'ютерів. Ці структури визначаються своєю здатністю до навчання та адаптації до змін, що робить їх незамінними в галузі штучного інтелекту.

1.2.1 Основні компоненти нейронних мереж:

Нейронні мережі складаються з базових компонентів, що моделюють нейрони та їх взаємодію[10]. Основні компоненти включають:

- Нейрони (Вузли): Це базові одиниці, які отримують вхідні сигнали, обробляють їх та генерують вихід. Кожен нейрон пов'язаний з вагами, які визначають вагомість вхідних сигналів.
- Зв'язки (Синапси): Це канали передачі сигналів між нейронами. Кожна зв'язка має вагу, яка впливає на сили взаємодії між нейронами.
- Шари: Нейронні мережі організовані у шари. Вхідний шар приймає вхідні дані, вихідний шар генерує результат, а проміжні шари виконують обчислення та витягують високорівневі ознаки.

Нейронні мережі вчаться з даних за допомогою процесу вивчення. Ваги зв'язків між нейронами поступово коригуються на основі введених даних та бажаного результату. Цей процес забезпечує адаптивність та здатність до самоорганізації мережі.

Кожен нейрон в нейронній мережі використовує функцію активації для вирішення, чи має нейрон передавати сигнал далі. Ця функція визначає вихідний сигнал на основі ваг і вхідних значень нейрона.

Механізм зворотного зв'язку використовується для корекції ваг у зворотньому напрямі. Після генерації вихідного сигналу порівнюється з бажаним результатом, і ваги нейронів коригуються відповідно.

1.2.2 Типи Нейронних Мереж

В основі Нейронної мережі полягає Персептрон.

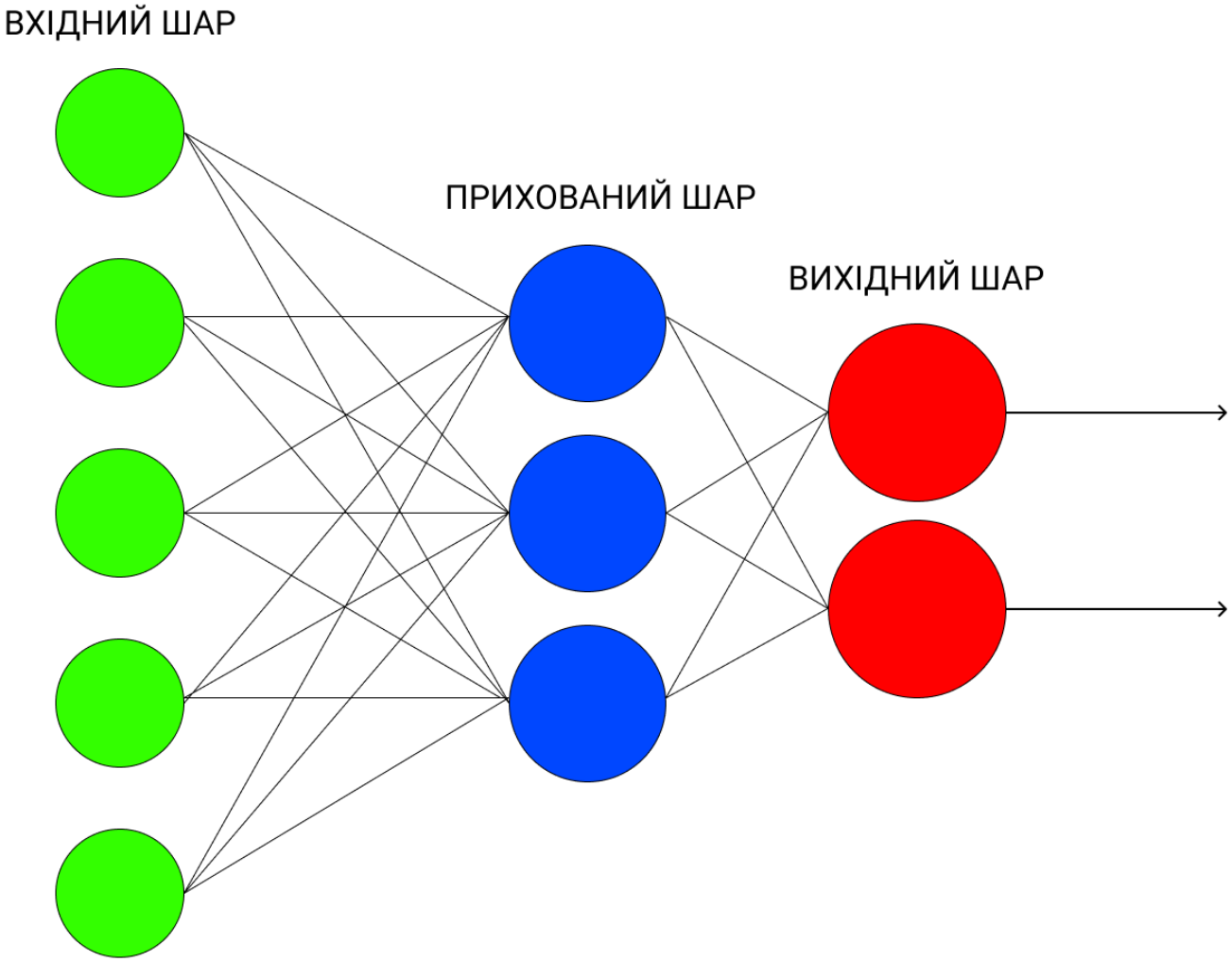


Рис. 2 — Персептрон

Персептрон – це проста форма нейронної мережі з одним шаром, що використовується для бінарної класифікації. Він є основою для більш складних мереж[10]. Такі як наприклад:

- Згорткові нейронні мережі (CNN):

Використовуються для обробки зображень та відео. Вони виявляють та враховують просторові особливості, роблячи їх ефективними для завдань комп'ютерного зору.

- Рекурентні нейронні мережі (RNN):

Застосовуються для роботи з послідовностями даних, такими як мовлення або

часові ряди. Вони можуть запам'ятовувати попередні стани, забезпечуючи здатність розуміти контекст.

Нейронні мережі використовуються в різноманітних галузях, включаючи медицину, фінанси, комп'ютерний зір, голосовий аналіз та інше. Їхні можливості безмежні, і з розвитком технологій цей інструмент продовжить трансформацію наукового прогресу та покращить технологічне різноманіття технологій.

РОЗДІЛ 2. ТЕОРЕТИЧНІ АСПЕКТИ ШТУЧНОГО ІНТЕЛЕКТУ В АВТОТРАНСПОРТІ

За останні кілька років автомобільна галузь переживає значні трансформації під впливом штучного інтелекту (ШІ) та автоматизованих систем. Введення ШІ в автомобільну індустрію відкриває нові можливості для підвищення рівня автономності та ефективності транспортних засобів[9]. Аналіз існуючих систем автоматизованого пілотування, такої як Tesla Autopilot, дозволить зрозуміти ключові технологічні рішення, що використовуються в сучасних автомобільних системах.

Одним з основних напрямків розвитку в сфері автомобільної галузі є розробка автономних транспортних засобів. Автономні автомобілі здатні переміщатися без участі водіїв, використовуючи розумні технології та інтелектуальні системи. ШІ в автомобільній галузі грає важливу роль у досягненні цих цілей.

Системи, що використовуються в автоматизованих автомобілях, базуються на комплексі електронних пристроїв, датчиків та алгоритмів ШІ. Наприклад, система Tesla Autopilot використовує набір датчиків, що дозволяють виявляти перешкоди на дорозі та контролювати рух автомобіля. Вона також використовує нейронні мережі для розпізнавання образів та вирішення проблем штучного зору.

Роль ШІ в автомобільній галузі включає багато аспектів, таких як обробка великого обсягу даних з різних датчиків, аналіз інформації про дорожню обстановку, прийняття рішень та контроль руху автомобіля. Часто використовуються нейронні мережі для вирішення цих завдань, оскільки вони добре справляються з обробкою образів та здатність до навчання з даних[10].

Окрім систем автоматизованого пілотування, ШІ використовується і для оптимізації транспортних систем. Наприклад, сучасні системи керування рухом можуть використовувати дані про рух транспортних засобів та дорожню ситуацію для розрахунку оптимальних маршрутів та регулювання сигналізації. ШІ також використовується для прогнозування популяційного руху автомобілів та

оптимізації графіків руху транспортних засобів.

Одним із ключових завдань, що стоїть перед ШІ в автомобільній галузі, є забезпечення безпеки та надійності автоматизованих систем. Запобігання аваріям та передбачення виникнення проблем є важливими чинниками у розвитку цих систем. Використання машинного навчання та аналітики даних допомагає вдалим чином аналізувати та передбачати небезпечні ситуації на дорозі, що підвищує загальний рівень безпеки.

Однак, впровадження ШІ в автомобільну галузь також поставило ряд викликів та проблем, які потребують уваги. Наприклад, низька ефективність алгоритмів розпізнавання для деяких умов дорожнього руху, проблеми з недостатньою визначеністю у розпізнаванні образів та інші наближення є частою проблемою для автоматизації автомобільної галузі[9].

Також, поставлення юридичних питань та недбалість з боку розробників можуть спричинити труднощі у реалізації автономних систем. З метою забезпечення безпеки на дорозі, необхідно розробляти політику та спеціальні правила, які регулюють використання автоматизованих транспортних засобів.

Загалом, використання штучного інтелекту у транспортній галузі відкриває нові можливості для покращення автономності та ефективності, але водночас ставить виклик перед інженерами та розробниками систем, пов'язаних із автомобільним транспортом. Розвиток і вдосконалення штучного інтелекту у цій галузі потребує постійного вдосконалення і зміцнення, що забезпечить розвиток безпечних та ефективних автономних транспортних засобів.

2.1 Огляд існуючих рішень

Tesla Autopilot

Tesla Autopilot[11] - це система автоматичного водіння, розроблена компанією Tesla Motors. Головна мета Autopilot полягає в тому, щоб зменшити необхідність активного керування автомобілем, навіть на довгих відрізках шляху або на швидкості на трасі. Вона забезпечує автопілотування автомобіля і дає

можливість водієві використовувати широкий спектр положень, які автомобіль може виконувати самостійно.

Tesla Autopilot[11] використовує штучний інтелект та комплексну систему датчиків для отримання інформації про дорожнє середовище. Це включає в себе передню камеру, радар, сенсори бокового зору та зонування. Ці датчики співпрацюють, щоб зорієнтуватися в просторі і зчитати дорожні знаки, визначити розмітку на дорозі та здійснити інші необхідні дії для безпечного водіння.

Принцип роботи Autopilot полягає в аналізі цієї отриманої інформації та автоматичному керуванні автомобілем відповідно до встановлених налаштувань водія. Технологія аналізує дані датчиків для визначення відстані до інших автомобілів, розпізнавання перешкод, зміни швидкості та керування в межах дорожньої розмітки.

Хоча Autopilot забезпечує автоматичне керування, водії повинні бути готові взяти керування на себе у випадках непередбачуваних ситуацій чи погіршення умов на дорозі. Tesla наголошує на важливості відповідального водіння та завжди залишати руки на кермі та бути готовими втрутитися у керування в будь-який момент.

Загалом, Tesla Autopilot відкриває нову еру автоматизованого керування, здійснюючи великий крок вперед у безпеці дорожнього руху та можливостях водіння автомобіля.

На даний час Tesla Autopilot[11] включає в себе ряд корисних функцій, які роблять водіння настільки простим та зручним, наскільки це можливо:

- Автоматичне керування по трасі: Autopilot здатний керувати автомобілем на трасах та автомагістралях, включаючи маневри зі зміненням полоси руху, а також активне прискорення та гальмування.
- Автоматична парковка: Tesla Autopilot може самостійно паркувати автомобіль, використовуючи передню та задню камери, а також сенсори на його боках.
- Система автостоянки: Коли водій виходить з автомобіля, система

Autopilot може викликати автоматичну стоянку, щоб автомобіль паркувався самостійно.

- Система автоматичної навігації: Tesla Autopilot може запам'ятати шляхи, які водій проїхав раніше. У разі повторного проїзду того ж шляху Autopilot візьме на себе керування, прокладе маршрут та виконає необхідні маневри.
- Віддалене керування: З допомогою мобільного додатку Tesla водій може керувати автомобілем, безпосередньо не знаходячись в автомобілі.

Загалом, Tesla Autopilot є досить ефективною системою автоматичного керування, яка покращує безпеку руху, зручність та задоволеність від водіння.

Переваги та недоліки системи Tesla Autopilot

- Покращена безпека та уникнення аварій: завдяки своїй високотехнологічній системі та сенсорам, система Autopilot може допомогти уникнути аварій, виявляючи можливі небезпеки на дорозі.
- Зменшення втоми та навантаження на водія: Виконуючи частину завдань за кермом, систма Autopilot може знизити навантаження на водія та допомогти уникнути втоми, яка може призвести до аварій.

Однак, також є деякі недоліки системи Tesla Autopilot, які повинні бути враховані:

- Надмірне покладанняся на технологію: деякі водії можуть стати занадто залежними від системи Autopilot та забути про дорогу, що може бути небезпечним та збільшити ризик аварій.
- Обмеження системи: система Autopilot не є повністю автономною та має певні обмеження, такі як неможливість керувати в певних умовах на дорозі.

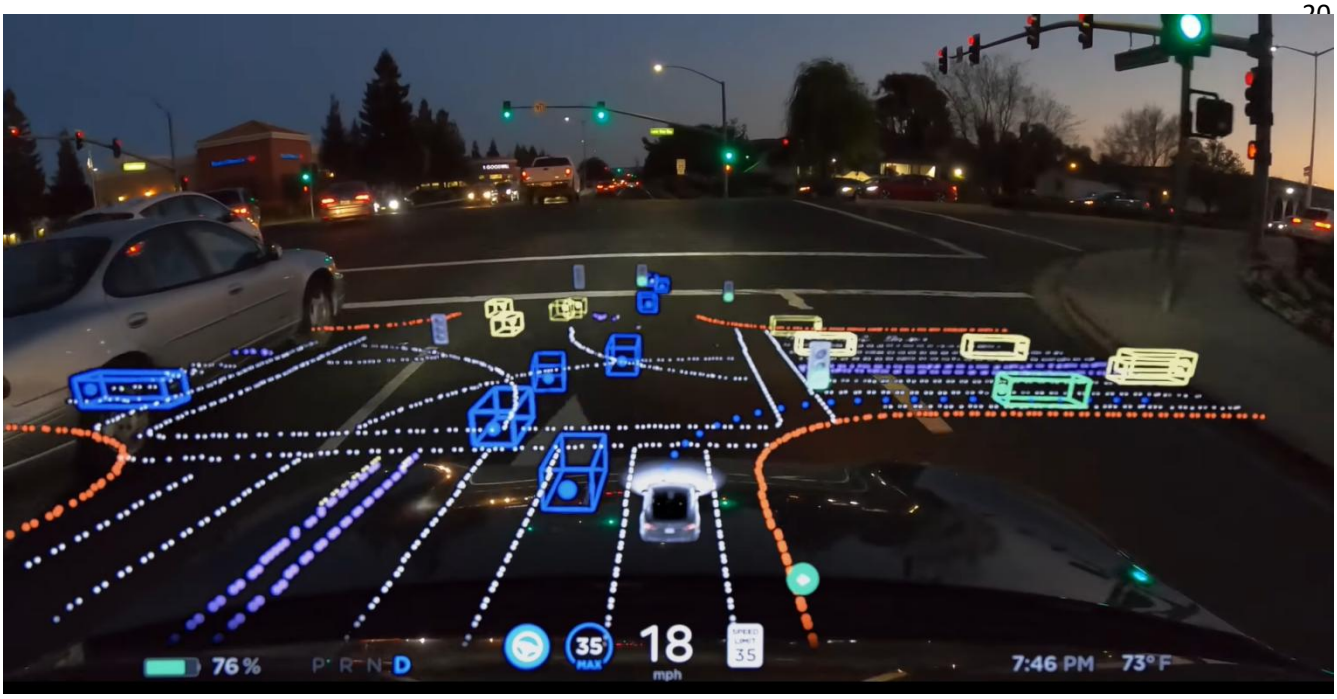


Рис. 3 — Орієнтування Tesla Autopilot на дорозі

Waymo Driver

Waymo Driver[12] є однією з передових систем глибокого навчання для автономного керування автомобілем. Розроблена компанією Waymo, яка є дочірнім підприємством Alphabet Inc. Ця технологія використовується у транспортних засобах, таких як автономні таксі та автобуси, з метою надання безпечного та ефективного транспортного обслуговування.

Робочий Принцип:

- Система відслідковування та сенсори:

  Waymo Driver використовує ряд сенсорів, таких як лідари, радари, камери та датчики, для отримання високоточної інформації про навколишнє середовище. Ці дані дозволяють системі точно розпізнавати об'єкти, дорожні знаки, пішоходів та інші транспортні засоби.

- Карти та локалізація:

  Алгоритми глибокого навчання використовуються для створення детальних карт місцевості та точної локалізації автомобіля на дорозі. Це дозволяє системі ефективно орієнтуватися в просторі та планувати

оптимальний маршрут.

- Нейронні мережі:

  Waymo Driver[12] базується на глибоких нейронних мережах для вивчання складних водійських задач, таких як взаємодія з іншими учасниками руху, безпечне зупинення та розпізнавання дорожніх знаків. Це дозволяє системі адаптуватися до різних дорожніх сценаріїв.

- Автономне управління:

  Система автономного управління Waymo Driver може взаємодіяти з іншими транспортними засобами та пасажирами, вирішувати важкі дорожні завдання, такі як обгін, перехід через перехрестя та управління ускладненими умовами руху.

- Навчання за допомогою сценаріїв:

  Система навчається на основі великої кількості сценаріїв, що дозволяє їй постійно вдосконалювати свої навички та адаптуватися до нових умов на дорозі.

Особливості та переваги:

- Висока точність:

  Waymo Driver визначається своєю високою точністю в розпізнаванні об'єктів та навігації в складних умовах дорожнього руху.

- Безпека та надійність:

- Система акцентується на безпеці, і надає водіям та пасажирам надійне та комфортне пересування.

- Адаптивність:

  Глибоке навчання дозволяє системі адаптуватися до різних дорожніх умов, забезпечуючи оптимальну ефективність.

- Постійне вдосконалення:

  Система має можливість навчатися на основі нових сценаріїв та вдосконалювати свої навички протягом часу.

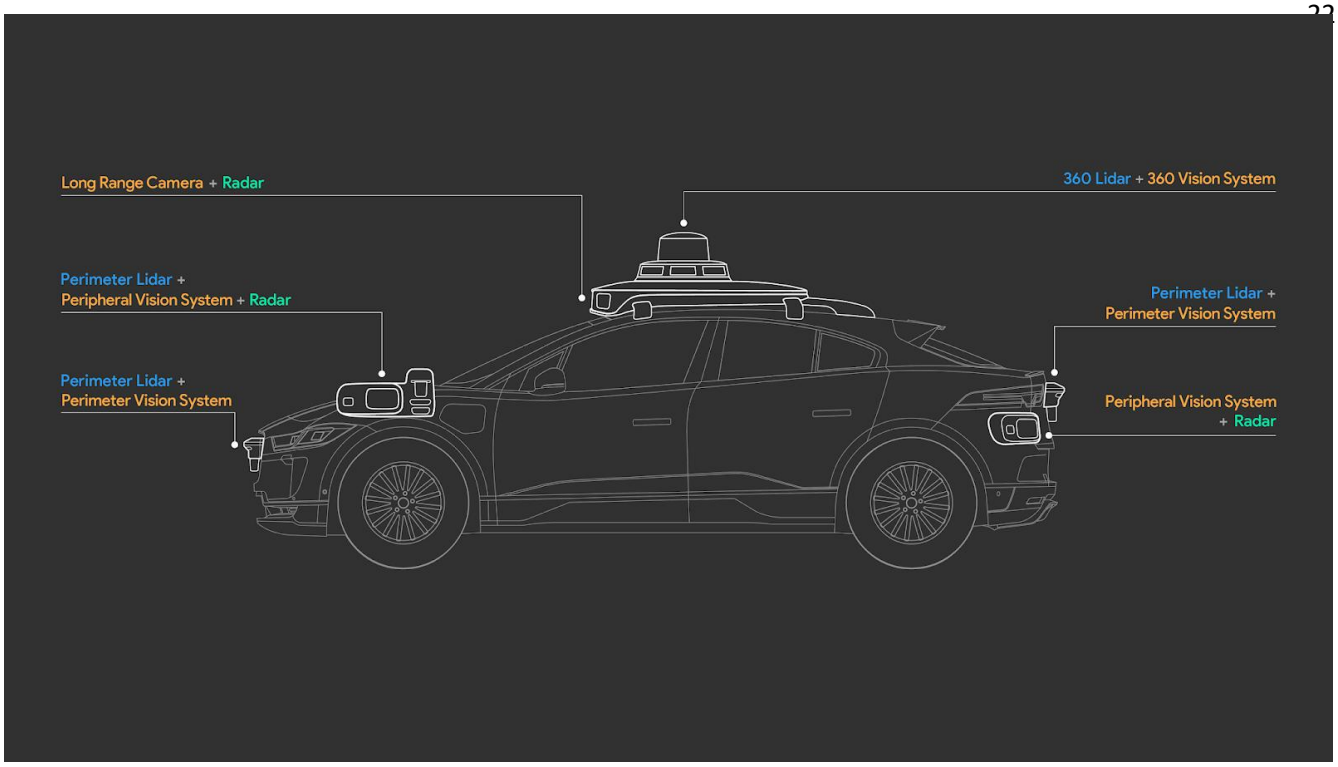


Рис. 4 — Оснащення автономного автомобіля з системою Waymo Driver

General Motors Super Cruise

Система General Motors Super Cruise[13] є однією з передових технологій автопілоту, розробленою компанією General Motors. Забезпечуючи високий рівень безпеки та комфорту, Super Cruise виводить автономне керування на новий рівень, дозволяючи водієві відпочивати, коли це можливо, і водночас забезпечуючи надзвичайно точне та безпечне автопілотування.

Основні Характеристики та Робочий Принцип:

- Адаптивний Круїз-Контроль:

Super Cruise включає в себе адаптивний круїз-контроль, який не тільки утримує задану швидкість, але й динамічно регулює швидкість та відстані до переднього автомобіля, забезпечуючи оптимальний рівень безпеки та ефективності.

- Відслідковування поглядом:

Система використовує відслідковування поглядом, щоб впевнитися, що водій уважно відстежує дорожню ситуацію. Камера над водійським місцем вивчає рухи очей та голови, і в разі виявлення відсутності уваги відбувається нагадування

водію взяти кермо у руки.

- Автоматичне управління по смузі:

Super Cruise може автоматично управляти автомобілем по смузі на автостраді. Він розпізнає дорожні знаки та розмітку, дозволяючи автомобілю точно слідувати заданому маршруту.

- Датчики та карти високої роздільності:

Для надання максимальної точності та безпеки, Super Cruise користується широким спектром датчиків, таких як лідари та радари, а також використовує високороздільні карти для деталізованої локалізації.

- Система відслідковування позначенням зміни полоси:

В разі, якщо автомобіль виходить за межі своєї смуги, система вчасно сповіщає водія та вживає відповідних заходів для відновлення коректного маршруту.

- Моніторинг зони автоматизованого управління:

Super Cruise працює тільки в заданих зонах автоматизованого управління, які картографуються та підтримуються для забезпечення безпечної експлуатації.

Переваги:

- Безпека та увага:

Super Cruise ставить безпеку на перший план, вживаючи заходів для виявлення та усунення можливих ризиків.

- Комфорт:

Водії можуть відчувати комфорт та розслабленість на довгих поїздках, дозволяючи Super Cruise взяти на себе контроль у відповідних умовах.

- Адаптивність та автономне керування:

Система розпізнає різні дорожні умови та адаптивно реагує, забезпечуючи автономне керування відповідно до задач та обставин.

- Ефективне використання технологій:

З використанням передових технологій, таких як датчики та високороздільні карти, Super Cruise створює оптимальні умови для ефективного та точного

автопілотування.

General Motors Super Cruise[13] визначається своєю інтелектуальністю та високим рівнем технічної досконалості, що робить його однією з передових систем автопілоту на ринку автомобільних технологій.

Рис. 5 — система Super Cruise

Audi Traffic Jam Pilot

Audi Traffic Jam Pilot[14] представляє сучасний погляд на автономне керування в умовах заторів. Розроблена компанією Audi AG, ця система вводить інноваційні технології, щоб забезпечити безпеку та комфорт водіїв у ситуаціях з інтенсивним трафіком.

Основні Характеристики та Робочий Принцип:

- Автономне керування в заторах:

Traffic Jam Pilot спеціалізується на автономному керуванні в умовах трафіку з низькою швидкістю. Система бере на себе відповідальність за керування

автомобілем в заторі, дозволяючи водієві відпочивати від монотонності руху.

- Керування рухом та гальмуванням:

Система взаємодіє з системою керування та гальмування автомобіля, забезпечуючи плавне та точне реагування на дорожні умови. Це включає управління швидкістю та безпечні взаємодії з іншими транспортними засобами.

- Відслідковування позначенням зміни полоси:

Traffic Jam Pilot вміє розпізнавати дорожні знаки та маркування, дозволяючи автомобілю впевнено утримуватися в межах своєї смуги та уникати конфліктів з іншими учасниками руху.

- Візуальне та акустичне сповіщення водію:

Система взаємодіє з водієм за допомогою візуальних та акустичних сигналів. Вона надсилає сповіщення та вказівки водієві для підтримання уваги та готовності взяти керування власними руками при потребі.

- Водійська контрольна функція:

Хоча Traffic Jam Pilot дозволяє автономне керування, водій повинен залишатися готовим прийняти керування в будь-який момент. Система виявляє ступінь уваги водія та може запросити його взяти керування, якщо це необхідно.

- Датчики та камери для навігації:

Traffic Jam Pilot використовує вбудовані датчики, лідари та камери для постійного моніторингу оточуючого середовища. Це дозволяє системі точно реагувати на зміни в трафіку та дорожніх умовах.

Переваги:

- Зменшення стресу в заторах:

Traffic Jam Pilot дозволяє водіям ефективно використовувати час у заторі для відпочинку та релаксації, сприяючи зменшенню стресу.

- Безпека та персоналізоване сповіщення:

Система розрізняє ступінь уваги водія та взаємодіє із сповіщеннями відповідно до потреб безпеки та комфорту.

- Адаптивність до дорожних ситуацій:

Traffic Jam Pilot[14] ефективно адаптується до різних дорожніх умов, забезпечуючи безпечне та ефективне керування в заторі.

Audi Traffic Jam Pilot визначається своєю здатністю допомагати водіям у заторах, дозволяючи їм насолоджуватися безпечною та комфортною поїздкою в умовах інтенсивного трафіку.

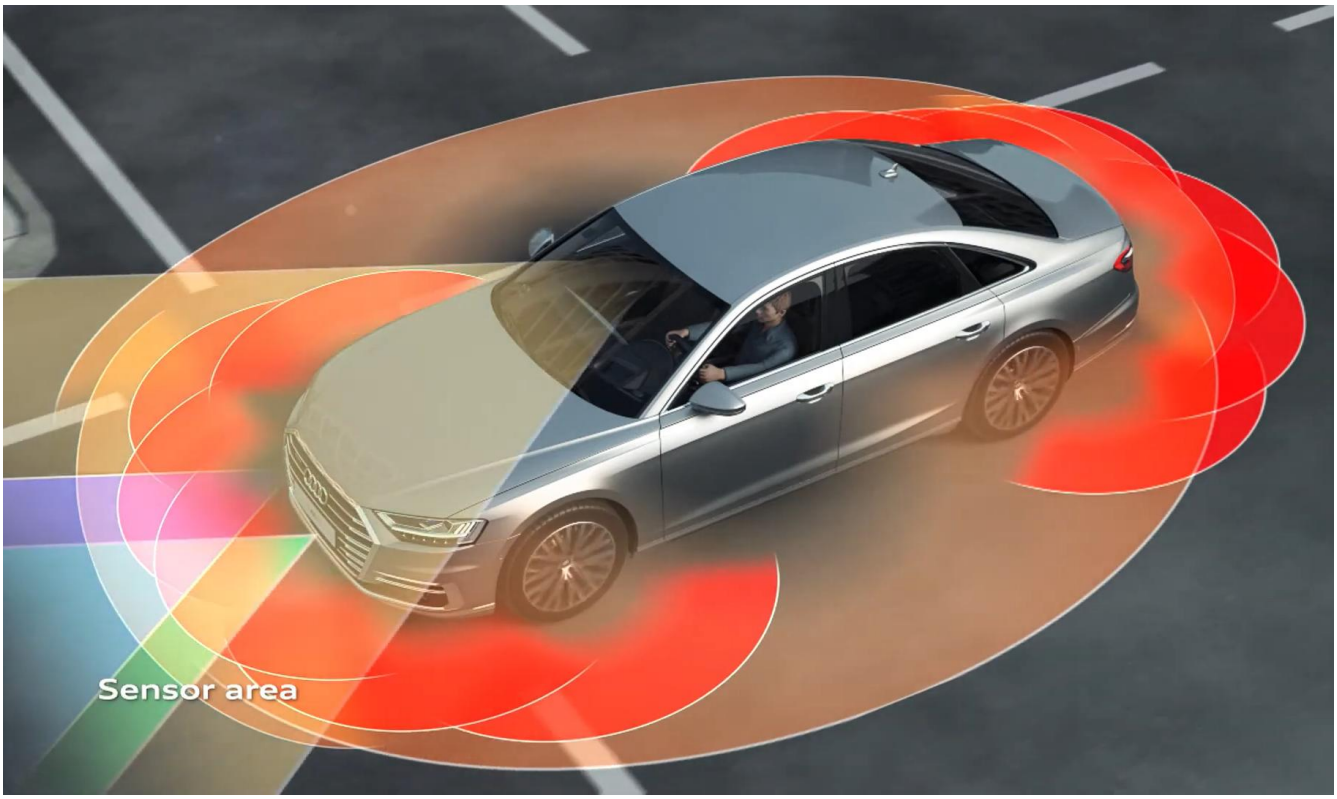


Рис. 6 — Поле зору Audi Traffic Jam Pilot

Nissan ProPILOT

Nissan ProPILOT[15] представляє сучасні технології автопілоту, розроблені компанією Nissan Motor Co., Ltd. Ця система призначена для полегшення керування автомобілем та створення безпечного та комфортного досвіду водіння, особливо в умовах загального трафіку та на довгих поїздках.

Основні характеристики та робочий принцип:

- Адаптивний Круїз-Контроль:

ProPILOT включає адаптивний круїз-контроль, який дозволяє автомобілю автоматично утримувати безпечну відстань до інших транспортних засобів на шляху. Система може адаптувати швидкість автомобіля в залежності від обраної

відстані та швидкості руху.

- Автоматичне управління по смузі:

Система дозволяє автомобілю утримуватися в межах своєї смуги руху. Вона використовує камери та сенсори для визначення розмітки та руху по дорозі.

- Активне кермове втручання:

ProPILOT може здійснювати активне кермове втручання для утримання автомобіля в центрі смуги. Це дозволяє системі реагувати на невеликі зміни в русі та тримати автомобіль стабільним на дорозі.

- Система відслідковування водія:

Датчики водіння виявляють ступінь уваги водія. Якщо водій не виявляє активності або його увага розсіюється, система може відправити сигнали та сповіщення для залучення уваги водія.

- Управління прискоренням та гальмуванням:

ProPILOT автоматично управляє прискоренням та гальмуванням відповідно до руху на дорозі та обраного водієм режиму.

- Технологія зонального автопілоту:

Система розрізняє області, де можливе використання автопілоту, та активує його відповідно до можливостей та безпекових стандартів.

Переваги:

- Зручність та комфорт:

ProPILOT робить водіння більш комфортним та менше втомлюючим, зокрема в умовах міського трафіку та на автострадах.

- Безпека та мінімізація ризиків:

Адаптивність системи та її можливості активно реагувати на рух та увагу водія зменшують ризики аварій та інцидентів.

- Ефективне використання палива:

Керування швидкістю та прискоренням здійснюється з урахуванням ефективного використання палива, що призводить до економії пального.

Nissan ProPILOT[15] визначається своєю здатністю спростити водіння та

надати допомогу водієві, роблячи поїздки більш зручними та безпечними.

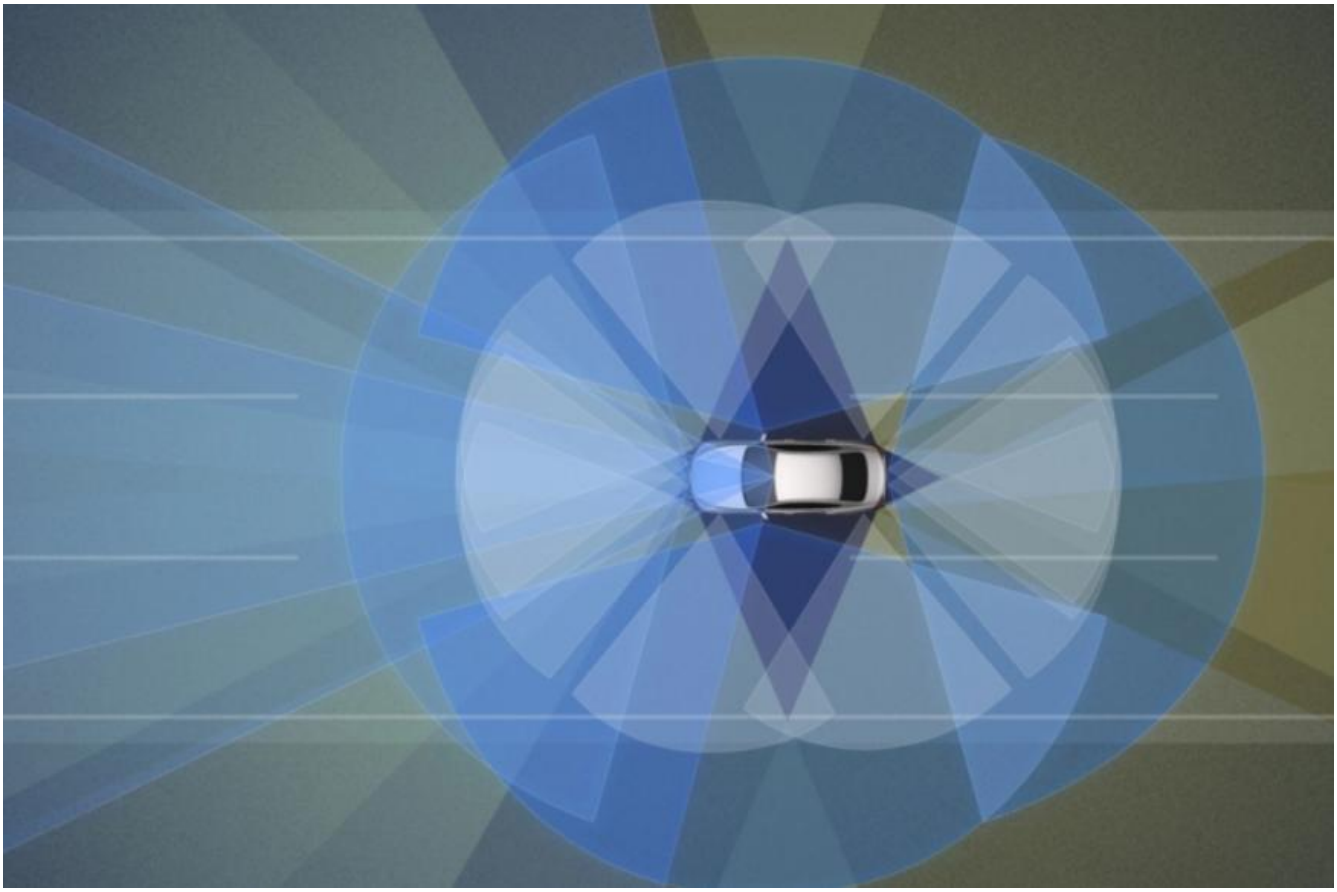


Рис. 7 — зона покриття датчиками системи Nissan ProPILOT

2.2 Етичні та юридичні аспекти автономних систем у транспорті

Автономні системи у транспорті, такі як автономні автомобілі, породжують численні етичні та юридичні виклики[9], оскільки їхнє впровадження може вплинути на безпеку, приватність, трудові відносини та соціокультурні аспекти суспільства.

Ключові аспекти:

- Етичні аспекти

Безпека перед усім:

Головною етичною вимогою є забезпечення безпеки для всіх учасників дорожнього руху. Автономні системи повинні приділяти велику увагу розробці та вдосконаленню алгоритмів, що максимізують безпеку пасажирів, пішоходів та інших учасників.

Рішення важких етичних ситуацій:

Автономні системи повинні бути запрограмовані для вирішення етично навантажених ситуацій, таких як вибір між рятуванням життя водія та інших учасників дорожнього руху.

Прозорість та відповідальність:

Розробники та виробники повинні забезпечувати прозорість у функціонуванні алгоритмів, надавати можливість розуміти прийняття рішень системою та нести відповідальність за можливі невдачі.

- Юридичні аспекти

Нормативне регулювання:

Потрібне створення та удосконалення законів та стандартів, які визначають правила використання автономних систем на дорогах. Це включає в себе визначення відповідальності в разі аварій та порушень.

Приватність та захист даних:

З використанням сенсорів та збору великої кількості даних, необхідно розробляти закони, що гарантують конфіденційність та захист особистої інформації користувачів.

Стандарти виробництва та безпеки:

Важливо встановити стандарти виробництва, які визначають якість та безпеку автономних систем. Це допоможе забезпечити виробників виконанням обов'язкових вимог[9].

Ліцензії та сертифікація:

Водії та виробники можуть вимагати спеціальних ліцензій та сертифікацій для використання та виробництва автономних транспортних засобів.

Адаптація правил руху:

Закони та правила дорожнього руху можуть потребувати адаптації для врахування особливостей автономного руху.

Відповідальність та страхування:

Необхідно розробляти механізми визначення відповідальності в разі аварій та визначення страхових правил для власників та виробників автономних систем.

Забезпечення балансу між інноваціями в автономних системах та збереженням етичних та юридичних стандартів є важливим завданням для ефективного впровадження цієї технології у транспорті.

2.3 Допоміжні системи штучного інтелекту орієнтування на дорозі

Штучний інтелект (ШІ) використовує різноманітні системи для орієнтації на дорозі, забезпечення безпеки та ефективного керування транспортними засобами[5]. Ці інтелектуальні системи, інтегровані в транспортні засоби, розвиваються для досягнення високого рівня автономії та покращення здатностей транспортних систем. Розглянемо деякі з ключових систем, які забезпечують роботу автономних транспортних засобів:

1. Системи комп'ютерного зорового сприйняття (Computer Vision):

Системи комп'ютерного зорового сприйняття використовують камери та сенсори для отримання візуальної інформації з оточуючого середовища. Ці системи дозволяють транспортним засобам розпізнавати дорожні знаки, інші транспортні засоби, пішоходів та інші об'єкти, надаючи повний образ дорожньої обстановки.

Рис. 7 — Computer Vision

2. Лідар (Lidar):

Лідар (Lidar) — це технологія вимірювання відстаней та визначення глибини, яка використовує лазерне випромінювання. Термін "Лідар" походить від слів "light" (світло) та "radar" (радар). Система Лідару використовує лазер для висилання коротких імпульсів світла в напрямленому промені, які відбиваються від поверхні об'єктів.

Основні компоненти системи Лідару включають:

- Лазерний випромінювач: генерує короткі лазерні імпульси, які висилаються в напрямленому промені.
- Детектор (Фотодіод): реєструє відбите світло від об'єктів та вимірює час, який затрачає світло на пройдення туди і назад.
- Гідро-Дзеркало (Поштовховий дзеркальний механізм): направляє промінь лазера у різні напрямки, дозволяючи сканувати оточуючий простір.
- Електроніка та комп'ютери: обробляють отримані дані і перетворюють їх у тривимірну карту оточуючого простору.

Лідар може вимірювати відстані з високою точністю, а також визначати форму та рельєф об'єктів. Використовується в різних галузях, таких як автономні автомобілі, картографія, геодезія, метеорологія та інші, де точність вимірювань є критично важливою. У сфері автономних транспортних засобів, Лідар використовується для створення точної тривимірної карти дороги та виявлення оточуючих об'єктів, що допомагає автомобілю безпечно та ефективно орієнтуватися на дорозі.

Рис. 8 — Lidar

3. Системи GPS (Global Positioning System):

GPS (Global Positioning System) – це система глобального позиціонування, яка використовує супутникову навігацію для визначення географічного положення та часу на землі в будь-якій точці планети. GPS включає в себе сітку супутників, приймачі та бодайні прилади.

Основні компоненти системи GPS:

- Супутники GPS: глобальна мережа штучних супутників, які обертаються навколо Землі. Кожен супутник висилає радіосигнали, що містять інформацію про своє положення та час.

- Приймач GPS: пристрій, який приймає сигнали від супутників та використовує їх для визначення свого географічного положення. Приймачі можуть бути вбудованими в смартфони, автомобільні навігатори, фотокамери, геодезичні інструменти тощо.
- Контрольні пункти та базові станції: Для забезпечення точності системи GPS, інформація від супутників також періодично коригується від контрольних пунктів та базових станцій на Землі.

Принцип роботи полягає в тому, що приймач GPS отримує сигнали від мінімум чотирьох супутників, розташованих у різних точках небесної сфери. За допомогою вимірювань часу, який пройшов від відправлення сигналу до його отримання приймачем, система може визначити точне положення приймача у тривимірному просторі.

GPS застосовується в різних галузях, таких як автомобільна навігація, логістика, картографія, рятувальні операції та інші.

4. Радар:

Радар (радіолокаційний детектор) - це електромагнітний прилад, який використовує радіовили з короткими довжинами хвиль для визначення розташування, відстані, швидкості або напрямку об'єктів, таких як літаки, кораблі, автомобілі, погодні умови і т. д. Технологія радару застосовується в різних галузях, включаючи оборонну промисловість, авіацію, судноплавство, метеорологію, астрономію і цивільний сектор.

Основні компоненти радару включають в себе:

- Трансмітер: Відповідає за генерацію електромагнітних хвиль та їх випромінення в простір у формі радіосигналів.
- Антена: Приймає випромінені сигнали, виконує обробку та фокусування, і відбирає відповідні відбиті сигнали від об'єктів.
- Приймач: Приймає сигнали, відбиті від об'єктів, і генерує електричний сигнал, який може бути оброблений для визначення відстані, швидкості, напрямку тощо.

- Дисплей та обробник сигналів: Виводить інформацію про виявлені об'єкти, а також відповідає за обробку сигналів для отримання потрібної інформації.

Радари можуть використовуватись для виявлення рухливих об'єктів у повітрі, на морі або на землі, а також для визначення характеристик атмосферних явищ, таких як опади та бурі. У цивільному транспорті, наприклад, радари застосовуються в системах контролю повітряного руху та автомобільних системах допомоги водієві для виявлення перешкод та уникнення зіткнень. В оборонній галузі радари використовуються для виявлення та слідження потенційних загроз.

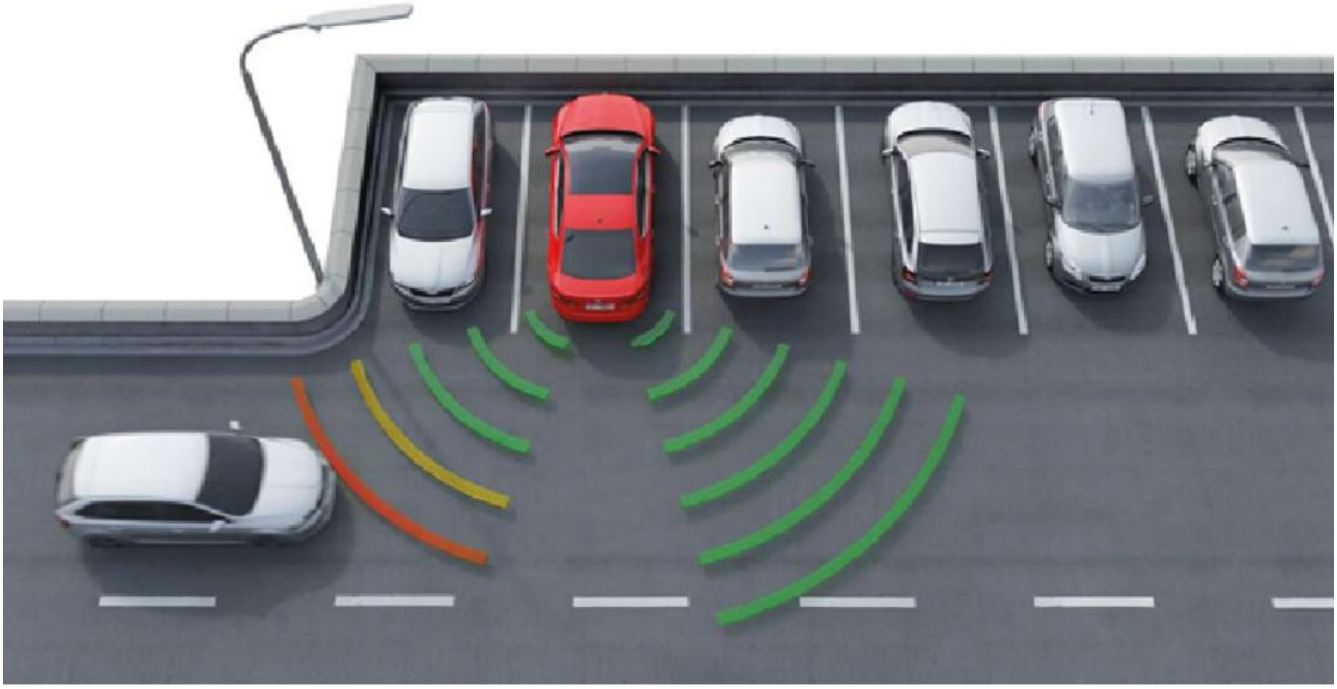


Рис. 9 — Радар

5. Датчики Інфрачервоного Сприйняття:

Датчики інфрачервоного сприйняття (ІЧ-датчики) - це пристрої, призначені для вимірювання і реєстрації інфрачервоного випромінювання, яке знаходиться в області електромагнітного спектру з довжинами хвиль більше, ніж видиме світло, але менше, ніж мікрохвильове випромінювання. ІЧ-випромінювання виникає внаслідок теплового випромінювання об'єктів, і такі датчики можуть

використовуватися для визначення температури об'єктів та їх розташування.

Основні компоненти датчиків інфрачервоного сприйняття:

- Фотодетектор: Це чутливий елемент, який перетворює інфрачервоне випромінювання на електричний сигнал.
- Оптична система: Лінзи або оптичні фільтри, які фокусують інфрачервоне випромінювання на фотодетекторі для збору сигналу.
- Електроніка обробки сигналів: Відповідає за обробку електричного сигналу, отриманого від фотодетектора, для визначення температури або інших характеристик об'єкта.
- Виведення результатів: Інформація про температуру або інші параметри може бути виведена на дисплей або передана до системи керування.

Датчики інфрачервоного сприйняття широко використовуються в різних галузях, таких як промисловість, безпека, медицина, автомобільна промисловість та інші. Вони можуть застосовуватися для виявлення об'єктів в умовах обмеженої видимості, вимірювання температури об'єктів, а також в системах виявлення руху та безпеки.

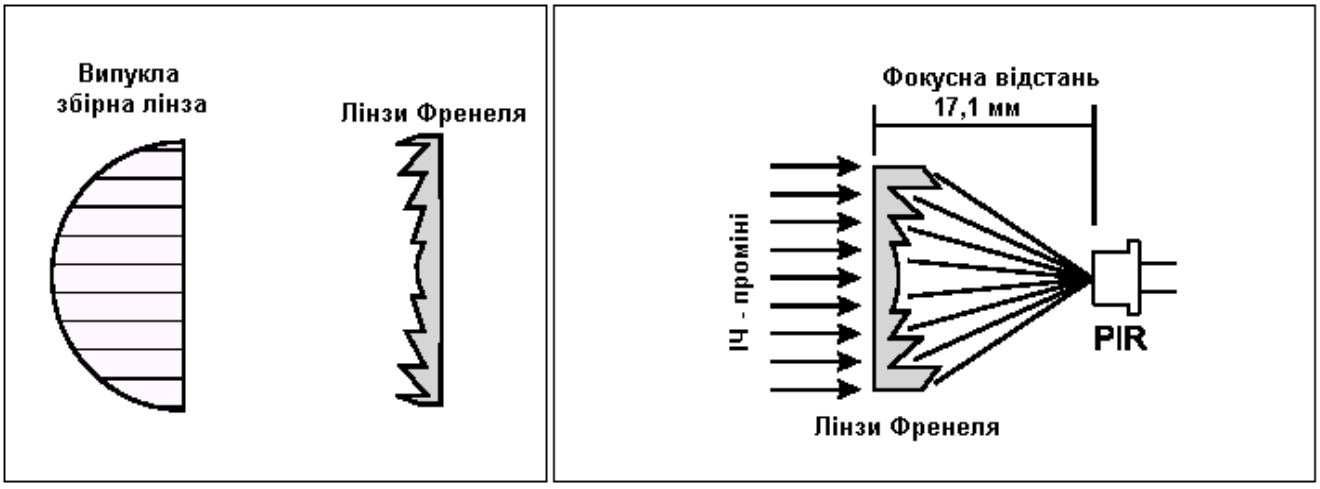


Рис. 10 — Датчики інфрачервоного сприйняття

6. Парктронік

Парктронік (англ. Parking sensor) — це система, яка використовується в автомобілях для допомоги водієві при паркуванні. Основна мета парктроніка —

уникнення зіткнень або пошкоджень автомобіля під час паркування в обмежених просторах.

Основні компоненти парктроніка включають в себе:

- Датчики: Зазвичай встановлюються в задній або передній частині автомобіля. Ці датчики виявляють наявність об'єктів навколо авто, а в деяких системах можуть бути розташовані в бамперах.
- Звуковий сигнал або дисплей: Інформує водія про відстань до ближчого об'єкта. Зазвичай, чим ближче об'єкт, тим частіше чути звуковий сигнал або виведено відповідні відомості на дисплей.
- Система керування: Опрацьовує інформацію від датчиків і визначає, наскільки близько знаходяться об'єкти. Деякі парктроніки також можуть включати функції автоматичного гальмування або автопаркування.

Парктронік допомагає водієві уникати зіткнень при паркуванні, особливо в тісних або заторованих місцях. Ця технологія стала стандартним обладнанням в багатьох сучасних автомобілях і допомагає зробити паркування більш безпечним і зручним процесом.

Ці розроблені системи стають невід'ємною частиною розумних транспортних засобів, що використовують ШІ для забезпечення безпеки та ефективного руху на дорогах[5].

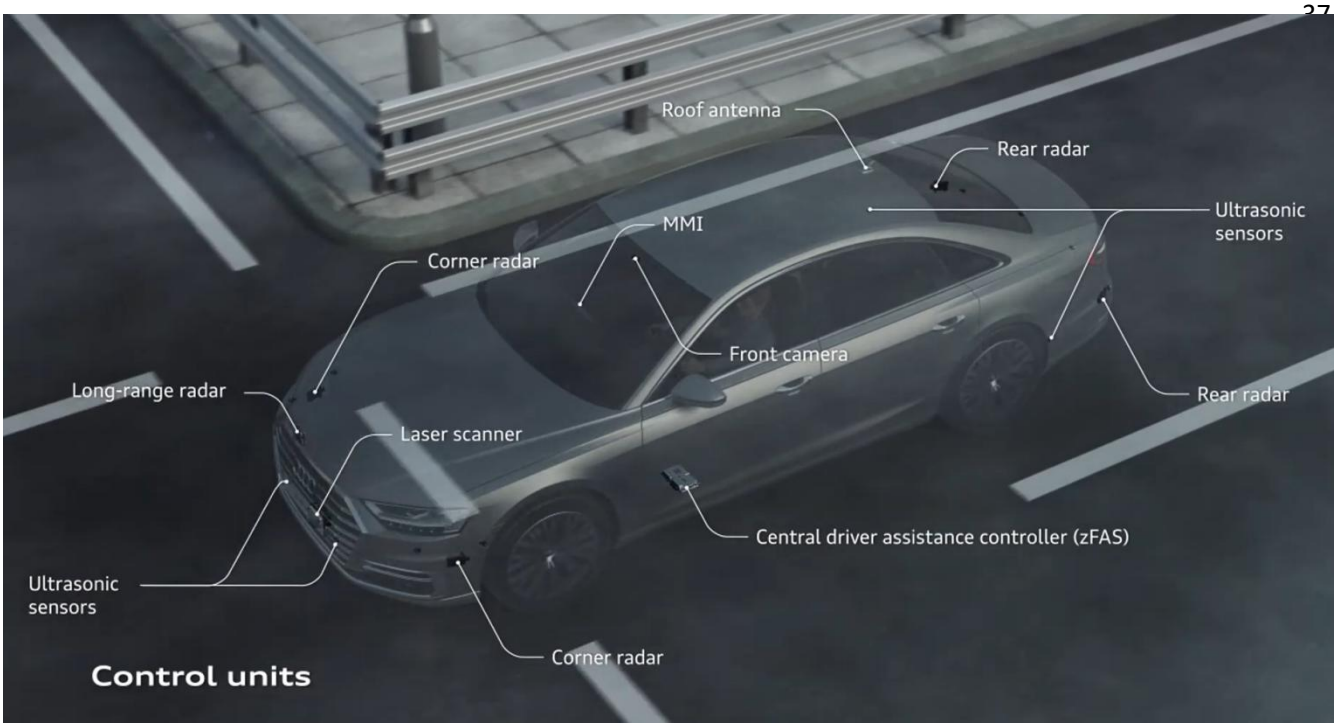


Рис. 11 — Схема розташувань допоміжних систем орієнтування автомобіля на дорозі

РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЗОВАНОГО ПІЛОТУВАННЯ

3.1 Середовище розробки та допоміжні бібліотеки

3.1.1 Python

У ході розробки даного проєкту використовувалась мова програмування Python[3], яка є однією з найпопулярніших та найвикористовуваніших мов програмування в галузі штучного інтелекту та нейромереж. Python має огромну кількість бібліотек, які дозволяють швидко та ефективно вирішувати складні задачі в галузі машинного навчання, такі як NEAT, Pygame. Python відомий своєю простотою та читабельністю коду, що значно спрощує процес розробки і підтримки програмного продукту. Безліч корисних інструментів та фреймворків, які доступні у мові, роблять Python чудовим вибором для роботи з штучним інтелектом та нейромережами.

3.1.2 PyCharm

Для комфортної розробки проекту було використано інтегроване середовище розробки (IDE) PyCharm. PyCharm забезпечує розширені можливості для розробки та підтримки проектів на мові Python. Це середовище надає розумні підказки, автоматичне завершення коду, вбудовану систему контролю версій та багато інших корисних функцій, що сприяють ефективному розвитку програмного продукту.

Використання Python та PyCharm у даній дипломній роботі дозволило ефективно реалізувати функціонал проекту та забезпечити його якість та надійність. Python призначений для швидкої розробки програмного забезпечення, має читабельний синтаксис та широкий спектр бібліотек, що дозволяють забезпечити потрібну функціональність. Вбудовані інструменти PyCharm сприяють збереженню історії змін коду, полегшують роботу зі змінними та функціями, а також допомагають швидко знаходити та виправляти помилки.

Таким чином, використання Python і PyCharm у даному проекті допомогли забезпечити комфортну та ефективну роботу, що в свою чергу позитивно вплинуло на якість та надійність розробленого програмного продукту.

3.1.3 NEAT

У проєкті використовувалася бібліотека NEAT (NeuroEvolution of Augmenting Topologies) для еволюції нейромереж[8]. `NEAT` є еволюційним алгоритмом, призначеним для навчання штучних нейромереж у режимі реального часу. Використання бібліотеки `NEAT` дозволяє автоматично створювати та оцінювати нейромережі, здійснювати їх еволюцію та вдосконалювати за допомогою генетичних алгоритмів.

`NEAT` вважається найліпшим інструментом для розв'язання задач, пов'язаних з навчанням машин та штучним інтелектом. В даному проєкті бібліотека `NEAT` дозволила здійснювати швидку та ефективну еволюцію нейромереж, що є важливим елементом даної області[5].

3.1.4 Pygame

Pygame — це бібліотека для створення симуляторів та графічних додатків на мові Python. Вона була використана для реалізації графічного інтерфейсу та візуалізації еволюції нейромереж в процесі симуляції.

Завдяки простоті використання та багатофункціональності Pygame, вдалося створити інтуїтивно зрозумілі симуляційні сцени, які допомагають покращити аналіз симуляції. Бібліотека Pygame надає широкі можливості для створення вікон, візуалізації графічних об'єктів, обробки подій та реалізації користувацького інтерфейсу. Це дозволяє зручно відображати процес еволюції нейромереж та забезпечило зрозумілу інтерактивну взаємодію з симуляцією.

3.2 Архітектува системи

NEAT (NeuroEvolution of Augmenting Topologies) - це алгоритм еволюції нейромереж, який став популярним у сфері штучного інтелекту та генетичних

алгоритмів. Його особливість полягає в можливості автоматичного зміщення і розширення архітектури нейромережі, що дозволяє еволюційно побудовані нейромережі орієнтуватися на більш складні задачі.

NEAT знайшов широке застосування у задачах, пов'язаних з навчанням нейромереж для керування агентами в симуляціях або в інших середовищах[6]. Завдяки своїй здатності пристосовуватися до змінних умов, NEAT може вирішувати різноманітні задачі і постійно покращувати свою продуктивність.

При використанні NEAT у сфері штучного інтелекту, вчитель (нейромережа) стикається з вихідними даними, вводить їх у нейромережу і отримує відповідні значення на виході. Ці вихідні значення порівнюються з очікуваними результатами, і на основі отриманої помилки нейромережа коригує свої внутрішні параметри. Така ітеративна процедура триває до досягнення оптимальних результатів.

NEAT був використаний для розробки систем штучного інтелекту, який керують поведінкою автономних агентів у симуляційному динамічному середовщі. Його гнучкість і здатність до самостійної адаптації дозволяють створити високоефективну систему штучного інтелекту, яка здатна пристосовуватися до зміни умов і постійно вдосконалювати свою функціональність.

3.2.1 Основні характеристики та концепції NEAT

- Еволюція топологій:

Одним із ключових аспектів NEAT є його здатність до еволюції самої топології нейромережі. Тобто, NEAT може генерувати нові шари, вузли і зв'язки або видаляти та модифікувати існуючі під час еволюції[5].

У початкових стадіях еволюції, коли нейромережа ще не ефективна, NEAT використовує прості структури без шарів і зв'язків. За допомогою генетичних операцій, таких як мутація або кросовер, можуть з'являтися нові шари, вузли і зв'язки в нейромережі. Це створює більше можливостей для передачі та обробки

інформації. Крім того, NEAT може видаляти непотрібні або зайві шари, вузли і зв'язки під час еволюції. Це дозволяє нейромережі адаптуватися до змінних вимог задачі, оптимізуючи топологію нейромережі для кращої ефективності та розв'язання складних завдань.

Така здатність NEAT до зміщення і розширення топології є одним із факторів, що допомагає покращити адаптивність та ефективність нейромережі під час еволюції.

- **Кодування генотипу**:

У NEAT агенти (нейромережі) представлені у вигляді генотипів, що за своєю суттю є послідовностями генів. Кожен ген відповідає за один елемент нейромережі: вузол, зв'язок або властивість вузла або зв'язку. Разом вони утворюють набір інструкцій, які описують топологію та параметри нейромережі.

Кожен вузол в генотипі містить інформацію про тип функції, яку він виконує, а також може містити параметри цієї функції. В залежності від типу функції нейрон може приймати декілька вхідних сигналів або не мати вхідних сигналів взагалі. Зв'язки між вузлами відображаються в генотипі за допомогою сполучень між ними. Кожен зв'язок містить в собі опис параметрів, таких як вага зв'язку та наявність або відсутність затримки у передачі сигналу.

Особливістю NEAT є те, що в генотипі нейромережі кожному вузлу і зв'язку присвоюється свій унікальний ідентифікатор. Така структура генотипів дозволяє з легкістю порівнювати і комбінувати різні генотипи нейромереж, що забезпечує більшу варіативність та швидкість еволюції[6].

Отже, генотипи в NEAT представляють собою послідовності генів, які кодують структуру та параметри агентів (нейромереж). Це дозволяє моделі застосовувати генетичні алгоритми до генерування та оптимізації нейромереж з мінімальною залежністю від параметрів та конкретної реалізації мережі[8].

- **Інновації та номера генів:**

Кожен генотип отримує унікальний номер, що визначає інновацію. Це дозволяє визначати еволюційні зміни у топології. Номер інновації присвоюється

кожному новому елементу опису нейромережі, такому як вузлу або зв'язку, якщо він є унікальним і до цього не зустрічався в раніше створених генотипах. Завдяки цьому інноваційні номери слугують як ідентифікатори для визначення однакових або подібних елементів нейромереж у різних генотипах.

Еволюційні зміни у топології визначаються згідно з інноваційними номерами. Порівняння інноваційних номерів в генотипах нейромереж дає змогу визначити, чи є дві нейромережі однакові або чи мають спільні елементи. Наприклад, якщо два генотипи мають однакові інноваційні номери для вузла і зв'язку, це означає, що вони мають еквівалентні елементи нейромережі. Такі збіги в інноваційних номерах допомагають виконувати кроссовер (схрещення) між генотипами та створювати нові комбінації у топології нейромережі. Таким чином, інноваційні номери генотипів допомагають ефективно керувати еволюційними змінами топології і стимулюють пошук нових та оптимальних структур нейромереж[5].

- **Виживання найкращих:**

Після оцінки придатності (фітнес-функції) кожного агента, кращі агенти відбираються для подальшої еволюції. Цей відбір здійснюється шляхом простого відбору кращих агентів, або застосовує складніші алгоритми відбору, такі як турнірний відбір або прохолодна зона.

Еволюція включає в себе дві основні операції: рекомбінацію та мутацію. Рекомбінація використовує концепцію кроссовера (схрещення), при якому вибрані батьківські агенти поєднують свої генотипи, щоб створити нового нащадка. Це дозволяє комбінувати корисні властивості генотипів, що може призвести до появи нових ефективних стратегій.

Мутація, з іншого боку, полягає у випадкових змінах генотипу, що сприяє створенню різноманітності серед агентів. Це дозволяє досліджувати нові рішення та уникати застрягання в локальних максимумах.

Після рекомбінації та мутації, нові нащадки оцінюються щодо їхньої придатності. Якщо результати їхньої оцінки кращі за попередніх агентів, то вони замінюють старих агентів у популяції[6]. Таким чином, еволюційні алгоритми

дозволяють покращувати популяцію в кожному поколінні шляхом поступового відбору, рекомбінації та мутації найкращих агентів.

- **Схрещування та мутація генотипів:**

Схрещування (рекомбінація) та мутація - це дві основні операції, які застосовуються до генотипів для створення нових потомків. Під час рекомбінації, обрані батьківські генотипи об'єднуються шляхом поєднання їхніх характеристик, щоб створити новий комбінацію генів, яка може мати нові корисні властивості. Цей процес може включати в себе обмін частин генотипів або зрізів генів з батьківських генотипів. З іншого боку, мутація полягає в зміні випадкових генів у генотипі, що веде до появи нових варіантів характеристик. Ці операції рекомбінації та мутації гарантують різноманітність в популяції та випадковість у пошуку ефективних рішень.

- **Врожденість:**

NEAT (NeuroEvolution of Augmenting Topologies) - це алгоритм, який використовує штучні нейронні мережі для розв'язування задач шляхом еволюції. Однією з цікавих ідей, яку впроваджено в NEAT, є поняття вродженості. Ця концепція означає, що кожна структура мережі має свій власний ідентифікатор. За допомогою цього ідентифікатора NEAT може легко порівнювати та спадкувати структури мереж між поколіннями. При розв'язуванні задач, які вимагають зміни структури мережі для досягнення оптимальної продуктивності, NEAT може застосовувати різні види мутацій та еволюційних операцій, що можуть призвести до зміни структури мережі. Однією з таких операцій є додавання нових нейронів чи зв'язків між нейронами. При цьому новостворені нейрони та зв'язки отримують власний ідентифікатор, що дозволяє NEAT простежувати їхню вродженість та спадкування між поколіннями. Таким чином, поняття вродженості в NEAT дозволяє зберігати та збільшувати різноманітність структур мереж у популяції за допомогою еволюційних операцій, що підвищує ефективність пошуку рішень в складних задачах[6].

3.3 Алгоритм системи

Реалізація симуляційного середовища для навчання автомобілів на основі алгоритму "Нейронової еволюції" (NEAT) [5] передбачає впровадження комплексної архітектури системи. Даний розділ присвячений детальному опису компонентів та їх взаємодії в рамках розробленої системи. Симулюляційне середовище складається з кількох ключових компонентів, які спільно працюють для досягнення поставленої мети - навчання автомобілів.

Один з центральних компонентів системи - це модель автомобіля. Ця модель представляє собою штучну нейронну мережу, яка використовується для прийняття рішень про рух автомобіля на основі поточного стану дороги та навколишнього середовища. Мережа отримує вхідні дані від різних сенсорів, таких як, відстань до перешкод (радар), швидкість та кут руху. Вихідні дані мережі використовуються для керування автомобілем, розрахунку кута повороту руля сили гальма та прискорення[10].

Щоб забезпечити генетичний алгоритм NEAT, у системі також присутня популяція автомобілів. Кожен автомобіль у популяції має свою власну нейронну мережу, що ініціалізується випадковим чином. Початкові мережі можуть бути простими та неефективними, але завдяки генетичним операціям, таким як мутація та схрещування, поступово розвиваються до більш складних та продуктивних структур. Це дозволяє системі еволюційно відібрати найкращі автомобілі та їхні мережі для подальшого навчання та використання[8].

Крім того, система має модель дороги та навколишнього середовища. Ця модель створює віртуальні об'єкти, які імітують дороги, які можуть впливати на рух автомобілів. Ці об'єкти були створені з різними рівнями складності, що дозволяє створювати різноманітні умови для навчання та тестування автомобілів.

Комунікація між всіма компонентами системи здійснюється за допомогою встановленого інтерфейсу. Мережі автомобілів отримують дані від радарів, обробляють їх та генерують керування. Система моделює рух автомобілів у віртуальному середовищі та оцінює продуктивність кожного автомобіля на основі

заданої метрики, такої як швидкість, безпека або енергоефективність.

Усі ці компоненти системи взаємодіють між собою впродовж кожної ітерації навчання. У кожній ітерації мережі автомобілів оновлюються, генеруються нові об'єкти в середовищі та проводиться оцінка продуктивності та збору даних для покращення навчання. Цей процес повторюється доти, доки не будуть досягнуті задані результати та вибрані найкращі мережі автомобілів[10].

- **Компонент Автомобіля:**

В рамках симуляції кожен автомобіль репрезентований об'єктом класу 'Car', який включає в себе необхідні дані для моделювання автомобіля - модель автомобіля, його характеристики руху та дані про оточуюче середовище.

Головною відповідальністю даного компонента є взаємодія автомобіля з дорожнім покриттям, визначення колізій та збір необхідної інформації про оточуюче середовище. Для досягнення цих цілей, компонент використовує радари, що допомагають зібрати дані про об'єкти, що перебувають в непосередній близькості до автомобіля. Загалом, компонент 'Car' виконує важливу роль в симуляції, дозволяючи автомобілю взаємодіяти з дорогою та оточуючими об'єктами.

Компонент Нейромережі:

Кожен автомобіль оснащений нейромережею, що несе на собі відповідальність за прийняття найвигідніших рішень щодо керування. Ця модель нейромережі базується на передовому алгоритмі NEAT (неоеволюційний алгоритм формування топології нейромережі). Шлях до максимальної продуктивності й оптимальності керування для кожного автомобіля прокладається через генетичний алгоритм, який створює власну нейромережу для кожного автомобіля, і потім оптимізує її для досягнення найкращих результатів на дорозі..

- **Компонент Середовища:**

Середовище, в якому рухаються автомобілі, реалістично відтворює віртуальну карту, яка визначає мережу доріг, різних за рівнями складності та більшою кількістю поворотів, що симулюють різку зміну дорожніх умов. Ці карти

завдяки використанню графічних об'єктів детально відображає реальні умови дорожнього руху. Графічна модель дозволяє автомобілю визначити свій маршрут, шляхом аналізування доступної інформації на карті, а також слідкувати за потенційними перешкодами та уникнути необхідності у спілкуванні з ними. Враховуючи ці дані, нейромережа здатна приймати оптимальні рішення щодо керування, забезпечуючи безпечну та ефективну їзду. Таке поєднання динамічної віртуальної карти та прогресивної нейромережі робить керування автомобілем насиченим, інтелектуальним та вірогідним відображенням реального світу.

- **Головна Логіка Симуляції:**

Головний цикл симуляції визначає взаємодію всіх компонентів. Для кожного кадру генеруються вхідні дані для нейромережі на основі отриманих даних з сенсорів автомобіля. Нейромережа приймає рішення, яке впливає на рух автомобіля в середовищі. Спостереження за виконанням завдань та обчислення фітнес-функцій також відбувається в цьому компоненті.

Ця інтегрована архітектура дозволяє системі не лише навчати автомобілі на основі алгоритму, але й ефективно моделювати їхню поведінку в складних умовах дорожнього руху. Кожен компонент виконує свою роль у створенні реалістичного та відтворюваного середовища для навчання.

Детальний опис цього процесу включає ще етапи:

1. Збір вхідних даних: Система отримує дані з сенсорів, таких як радари який надає інформацію про оточуюче середовище автомобіля.

2. Передача даних до нейромережі: Отримані дані передаються у вхідний шар нейромережі, де вони обробляються та аналізуються для прийняття рішень.

3. Прийняття рішень: Нейромережа визначає оптимальні дії, які автомобіль повинен здійснити в даний момент, враховуючи отримані дані та поставлені завдання.

4. Моделювання руху автомобіля: Враховуючи прийняте рішення, система оновлює внутрішні параметри автомобіля, такі як швидкість, напрямок руху, прискорення тощо, щоб відобразити рух автомобіля в середовищі.

5. Спостереження та оцінка: Система спостерігає за виконанням поставлених завдань, а також обчислює фітнес-функції, які використовуються для оцінки якості роботи автомобіля.

6. Повторення циклу: Цей процес повторюється протягом декількох кадрів в секунду, щоб забезпечити неперервну та динамічну симуляцію руху автомобіля в середовищі.

Така інтегрована архітектура дозволяє системі ефективно навчати та моделювати поведінку автомобіля в реальних дорожніх умовах, що веде до покращення безпеки та продуктивності дорожнього руху.

РОЗДІЛ 4. АПРОБАЦІЯ В СИМУЛЯЦІЙНОМУ СЕРЕДОВИЩІ

Даний розділ присвячений детальній апробації розробленої системи штучного інтелекту в симуляційному середовищі. З метою оцінки ефективності та визначення відповідності системи визначеним критеріям, були проведені експерименти та аналіз отриманих результатів.

4.1 Методика апробації

Для апробації системи використовувалася симуляційна модель, яка відтворює реальні умови автотранспорту. Вхідними даними для системи були визначені параметри дорожнього середовища, включаючи складність доріг та характеристики автомобілів.

Кожен автомобіль у симуляційному середовищі був обладнаний системою штучного інтелекту, розробленою за допомогою алгоритму "Нейронової еволюції" (NEAT). Кожен автомобіль мав можливість приймати рішення щодо зміни напрямку руху, швидкості та інших параметрів на основі внутрішніх сенсорів та зовнішніх умов.

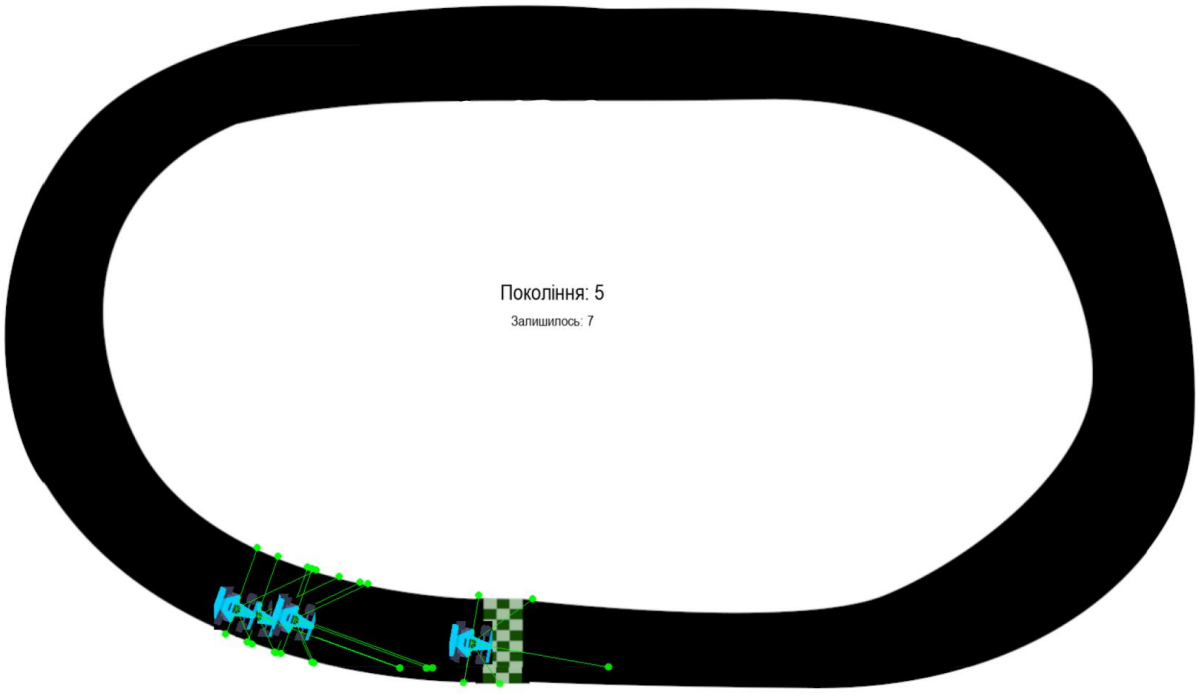


Рис. 12 — Тестування алгоритму на першому рівні складності

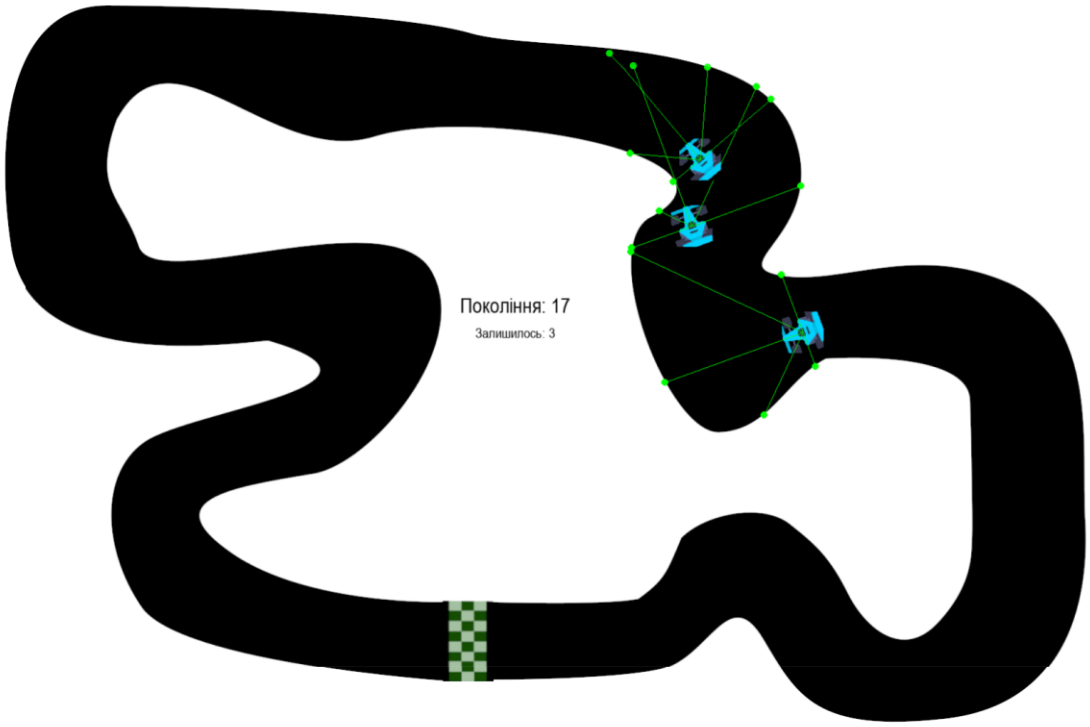


Рис. 13 — Тестування алгоритму на другому рівні складності

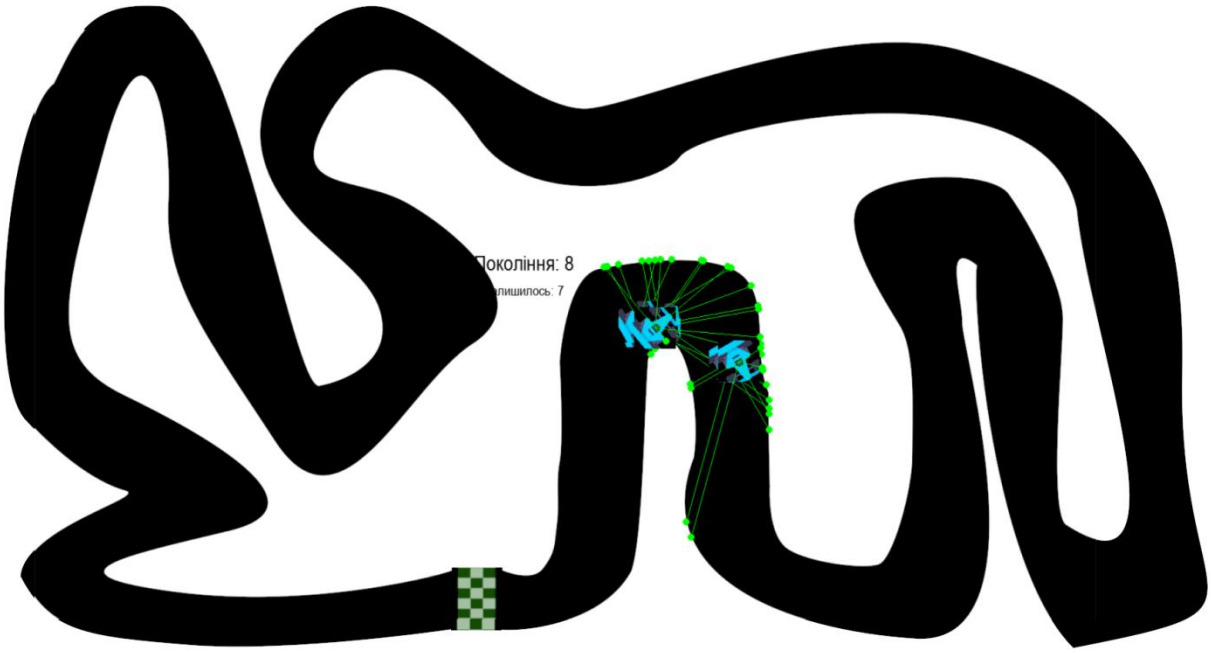


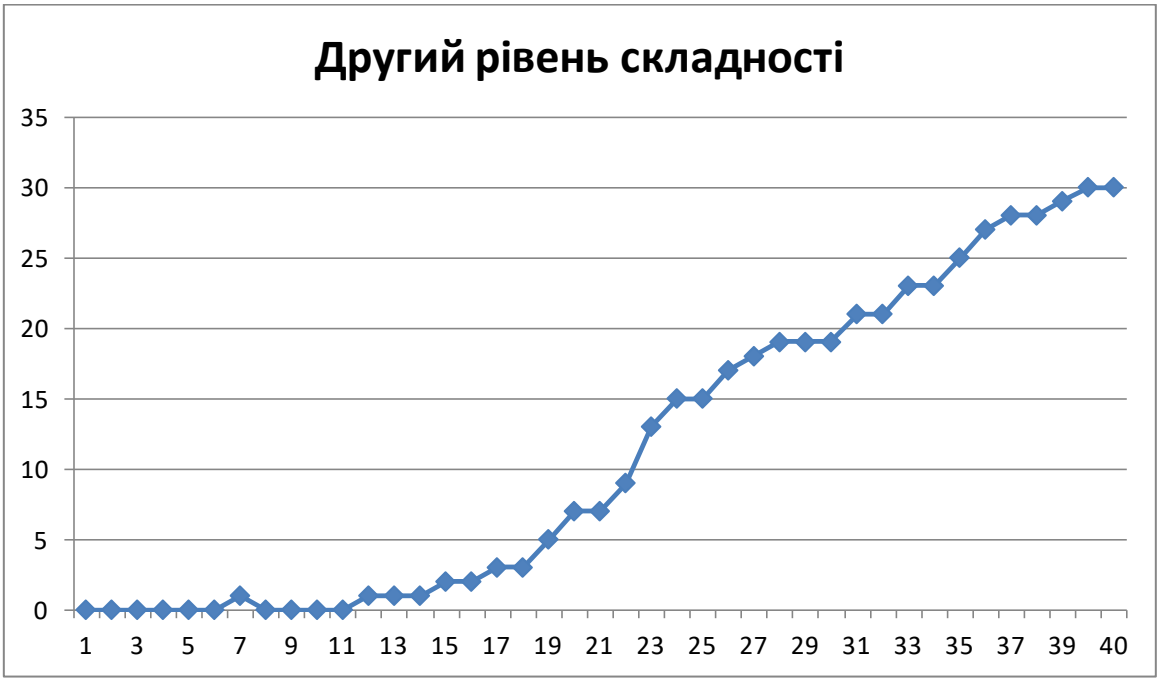
Рис. 14 — Тестування алгоритму на третьому рівні складності

4.2 Оцінка результатів

Оцінка ефективності системи в стимуляційному середовищі проводилася на основі результатів виживання на основі проходження трас автомобілів:

Було визначено кількість автомобілів, які успішно завершили маршрут без зіткнень:

Графік 1 — Виживання автомобілів на першому рівні складності (25 поколінь,30 автомобілів)



Графік 2 — Виживання автомобілів на другому рівні складності (40 поколінь,30 автомобілів)

Графік 3 — Виживання автомобілів на третьому рівні складності (70 поколінь,30 автомобілів)

4.3 Результати апробації

Проведені експерименти показали, що алгоритм NEAT, використаний у симуляційному середовищі для навчання автомобілів, проявив високу ефективність та добре справлявся із завданням оптимізації керування. Отримані дані свідчать про значний прогрес у розвитку автономних агентів, здатних адаптуватися до різноманітних умов та вирішувати складні завдання.

Зокрема, з'ясовано, що автомобілі, навчені з використанням NEAT, здатні дошкульно вчитися вирішувати завдання керування, оптимізуючи свої стратегії в реальному часі. Ефективність цього підходу проявилася в досягненні високого рівня автономії та самостійності в управлінні транспортними засобами в різних умовах руху.

Додатково, результати апробації свідчать про гнучкість та розширюваність симуляційної системи, яка може легко адаптуватися для вивчення та оптимізації

різних видів транспорту та різноманітних умов дорожнього руху.

У цілому, експериментальні результати підтверджують потенціал використання алгоритму NEAT у сфері навчання автономних систем управління автомобілями, що відкриває перспективи для подальших досліджень та розвитку цього підходу в області штучного інтелекту.

4.4 Перспективи та подальші дослідження

На основі результатів апробації можна визначити напрямки для подальших досліджень та вдосконалення системи. Зокрема, можливість оптимізації та розширення функціоналу системи становить ключовий аспект для подальшого розвитку. Для досягнення цього можна розглядати такі можливі вдосконалення:

1. Оптимізація алгоритмів навчання: Вдосконалення ефективності алгоритмів навчання, включаючи параметри NEAT, для ще швидшого та точнішого формування оптимальних стратегій управління автомобілями.

2. Розширення набору завдань: Додавання різноманітних сценаріїв та завдань для тренування автомобілів у більш широкому спектрі умов дорожнього руху.

3. Вдосконалення симуляційного середовища: Покращення фізичної моделі автомобіля та оточуючого середовища, щоб наблизити симуляцію до реальних умов.

4. Впровадження зворотного зв'язку: Розробка механізмів зворотного зв'язку для автомобілів, щоб система була здатна активно вчитися на основі результатів своїх дій.

У цілому, апробація системи в симуляційному середовищі була успішною і підтвердила високий рівень автономії та навчання системи за допомогою NEAT. Це відкриває шлях для подальшого вдосконалення та впровадження розробленої системи у реальні умови автотранспорту, що має великий потенціал у сфері розвитку автономних транспортних засобів.

ВИСНОВОК

В процесі виконання дипломної роботи було проведено аналіз предметної області, де детально вивчені основні аспекти штучного інтелекту, зокрема машинного та глибокого навчання, а також концепції навчання з підкріпленням. Розглянуті основні компоненти та типи нейронних мереж, що становлять основу для впровадження штучного інтелекту в автотранспорт.

В другому розділі проведений огляд існуючих рішень у галузі автономного транспорту, включаючи системи таких компаній, як Tesla, Waymo, General Motors, Audi та Nissan. Детально розглянуті основні аспекти штучного інтелекту, а також виникнення етичних та юридичних питань, пов'язаних з впровадженням автономних систем.

Третій розділ присвячений розробці та реалізації системи штучного інтелекту для автоматизованого пілотування. Описана архітектура системи, базові характеристики та концепції алгоритму NEAT. Розглянуто основні етапи алгоритму та визначено середовище розробки.

У розділі "Апробація в симуляційному середовищі" проведено тестування розробленої системи в симуляційному середовищі. Оцінено роботу системи з використанням алгоритму NEAT, визначено її ефективність та відповідність встановленим критеріям.

У висновках можна констатувати, що розроблено і реалізовано систему штучного інтелекту для автоматизованого пілотування, яка враховує сучасні тенденції в галузі та відповідає високим стандартам ефективності та безпеки. Робота включає важливий аспект етичних та юридичних питань, які виникають у контексті впровадження автономних систем у транспорті, а також відзначає необхідність подальших досліджень у цьому напрямку.

СПИСОК ЛІТЕРАТУРИ

1. "Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig, 2021. с.54 — 175.

2. "Reinforcement Learning: An Introduction" by Richard S. Sutton and Andrew G. Barto, 2018. с.35 — 68.

3. "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili, 2017. с.122 – – 157.

4. "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville, 2016. с.22 — 83.

5. "Neuroevolution: A Beginner's Guide" by Sebastian Risi and Kenneth O. Stanley, 2019. с.9 — 102.

6. "Evolutionary Algorithms for Solving Multi-Objective Problems" by Carlos A. Coello Coello, 2007. с.2 — 35.

7. "Machine Learning Yearning" by Andrew Ng, 2018. с.56 — 112.

8. "Neuro-Dynamic Programming" by Dimitri P. Bertsekas and John N. Tsitsiklis, 1996. с.48 — 65.

9. "Evolutionary Computation: A Unified Approach" by Kenneth De Jong, 2006. с.45 – – 56.

10. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, 2009. с.78 — 93.

11. Tesla Autopilot Official Website — 2023 — URL: https://www.tesla.com/support/autopilot

12.Waymo Driver Official Website — 2023 — URL: https://waymo.com/waymo-driver/

13. General Motors Super Cruise Official Website — 2023 — URL: https://www.chevrolet.com/super-cruise

14. Audi Traffic Jam Pilot Official Website — 2023 — URL: https://www.audi.ua

15. Nissan ProPILOT Official Website — 2023 — URL: https://www.nissan.ua/experience-nissan.html

ДОДАТОК А

```
import math
import sys

import neat
import pygame

# Constants

NEW_WIDTH = 1920
NEW_HEIGHT = 1080

CAR_SIZE_X = 60
CAR_SIZE_Y = 60

BORDER_COLOR = (255, 255, 255, 255)  # Color To Crash on Hit

current_generation = 0  # Generation counter


class Car:
    def __init__(self):
        # Load Car Sprite and Rotate
        self.sprite = pygame.image.load('car.png').convert()  # Convert Speeds Up A Lot
        self.sprite = pygame.transform.scale(self.sprite, (CAR_SIZE_X, CAR_SIZE_Y))
        self.rotated_sprite = self.sprite

        self.position = [830, 920]  # Starting Position
        self.angle = 0
        self.speed = 0

        self.speed_set = False  # Flag For Default Speed Later on

        self.center = [self.position[0] + CAR_SIZE_X / 2, self.position[1] + CAR_SIZE_Y
/ 2]  # Calculate Center

        self.radars = []  # List For Sensors / Radars
        self.drawing_radars = []  # Radars To Be Drawn

        self.alive = True  # Boolean To Check If Car is Crashed

        self.distance = 0  # Distance Driven
        self.time = 0  # Time Passed
```

```
        self.corners = []  # Initialize Corners

    def draw(self, screen):
        screen.blit(self.rotated_sprite, self.position)  # Draw Sprite
        self.draw_radar(screen)  # OPTIONAL FOR SENSORS

    def draw_radar(self, screen):
        # Optionally Draw All Sensors / Radars
        for radar in self.radars:
            position = radar[0]
            pygame.draw.line(screen, (0, 255, 0), self.center, position, 1)
            pygame.draw.circle(screen, (0, 255, 0), position, 5)

    def check_collision(self, game_map):
        self.alive = True
        for point in self.corners:
            # If Any Corner Touches Border Color -> Crash
            # Assumes Rectangle
            if game_map.get_at((int(point[0]), int(point[1]))) == BORDER_COLOR:
                self.alive = False
                break

    def check_radar(self, degree, game_map):
        length = 0
        x = int(self.center[0] + math.cos(math.radians(360 - (self.angle + degree))) *
length)
        y = int(self.center[1] + math.sin(math.radians(360 - (self.angle + degree))) *
length)

        # While We Don't Hit BORDER_COLOR AND length < 300 (just a max) -> go
further and further
        while not game_map.get_at((x, y)) == BORDER_COLOR and length < 300:
            length = length + 1
            x = int(self.center[0] + math.cos(math.radians(360 - (self.angle + degree))) *
length)
            y = int(self.center[1] + math.sin(math.radians(360 - (self.angle + degree))) *
length)

        # Calculate Distance To Border And Append To Radars List
        dist = int(math.sqrt(math.pow(x - self.center[0], 2) + math.pow(y - self.center[1],
2)))
        self.radars.append([(x, y), dist])
```

```
    def update(self, game_map):
        # Set The Speed To 20 For The First Time
        # Only When Having 4 Output Nodes With Speed Up and Down
        if not self.speed_set:
            self.speed = 20
            self.speed_set = True

        # Get Rotated Sprite And Move Into The Right X-Direction
        # Don't Let The Car Go Closer Than 20px To The Edge
        self.rotated_sprite = self.rotate_center(self.sprite, self.angle)
        self.position[0] += math.cos(math.radians(360 - self.angle)) * self.speed
        self.position[0] = max(self.position[0], 20)
        self.position[0] = min(self.position[0], NEW_WIDTH - 120)

        # Increase Distance and Time
        self.distance += self.speed
        self.time += 1

        # Same For Y-Position
        self.position[1] += math.sin(math.radians(360 - self.angle)) * self.speed
        self.position[1] = max(self.position[1], 20)
        self.position[1] = min(self.position[1], NEW_HEIGHT - 120)

        # Calculate New Center
        self.center = [int(self.position[0]) + CAR_SIZE_X / 2, int(self.position[1]) +
CAR_SIZE_Y / 2]

        # Calculate Four Corners
        # Length Is Half The Side
        length = 0.5 * CAR_SIZE_X
        left_top = [self.center[0] + math.cos(math.radians(360 - (self.angle + 30))) * length,
                    self.center[1] + math.sin(math.radians(360 - (self.angle + 30))) * length]
        right_top = [self.center[0] + math.cos(math.radians(360 - (self.angle + 150))) *
length,
                     self.center[1] + math.sin(math.radians(360 - (self.angle + 150))) * length]
        left_bottom = [self.center[0] + math.cos(math.radians(360 - (self.angle + 210))) *
length,
                       self.center[1] + math.sin(math.radians(360 - (self.angle + 210))) * length]
        right_bottom = [self.center[0] + math.cos(math.radians(360 - (self.angle + 330))) *
length,
                        self.center[1] + math.sin(math.radians(360 - (self.angle + 330))) * length]
        self.corners = [left_top, right_top, left_bottom, right_bottom]
```

```
        # Check Collisions And Clear Radars
        self.check_collision(game_map)
        self.radars.clear()

        # From -90 To 120 With Step-Size 45 Check Radar
        for d in range(-90, 120, 45):
            self.check_radar(d, game_map)

    def get_data(self):
        # Get Distances To Border
        radars = self.radars
        return_values = [0, 0, 0, 0, 0]
        for i, radar in enumerate(radars):
            return_values[i] = int(radar[1] / 30)

        return return_values

    def is_alive(self):
        # Basic Alive Function
        return self.alive

    def get_reward(self):
        # return self.distance / 50.0
        return self.distance / (CAR_SIZE_X / 2)

    @staticmethod
    def rotate_center(image, angle):
        # Rotate The Rectangle
        rectangle = image.get_rect()
        rotated_image = pygame.transform.rotate(image, angle)
        rotated_rectangle = rectangle.copy()
        rotated_rectangle.center = rotated_image.get_rect().center
        rotated_image = rotated_image.subsurface(rotated_rectangle).copy()
        return rotated_image


def run_simulation(genomes, config1):
    # Empty Collections For Nets and Cars
    nets = []
    cars = []

    # Initialize PyGame And The Display
```

```
pygame.init()
screen = pygame.display.set_mode((NEW_WIDTH, NEW_HEIGHT))

# For All Genomes Passed Create A New Neural Network
for i, g in genomes:
    net = neat.nn.FeedForwardNetwork.create(g, config1)
    nets.append(net)
    g.fitness = 0

    cars.append(Car())

# Clock Settings
# Font Settings & Loading Map
clock = pygame.time.Clock()
generation_font = pygame.font.SysFont("Arial", 30)
alive_font = pygame.font.SysFont("Arial", 20)
game_map = pygame.image.load('map.png').convert()
game_map = pygame.transform.scale(game_map, (NEW_WIDTH, NEW_HEIGHT))

global current_generation
current_generation += 1

# Simple Counter To Roughly Limit Time (Not Good Practice)
counter = 0

while True:
    # Exit On Quit Event
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit(0)

    # For Each Car Get The Acton It Takes
    for i, car in enumerate(cars):
        output = nets[i].activate(car.get_data())
        choice = output.index(max(output))
        if choice == 0:
            car.angle += 10  # Left
        elif choice == 1:
            car.angle -= 10  # Right
        elif choice == 2:
            if car.speed - 2 >= 12:
                car.speed -= 2  # Slow Down
        else:
```

```
            car.speed += 2  # Speed Up

        # Check If Car Is Still Alive
        # Increase Fitness If Yes And Break Loop If Not
        still_alive = 0
        for i, car in enumerate(cars):
            if car.is_alive():
                still_alive += 1
                car.update(game_map)
                genomes[i][1].fitness += car.get_reward()

        if still_alive == 0:
            break

        counter += 1
        if counter == 30 * 40:  # Stop After About 20 Seconds
            break

        # Draw Map And All Cars That Are Alive
        screen.blit(game_map, (0, 0))
        for car in cars:
            if car.is_alive():
                car.draw(screen)

        # Display Info
        text = generation_font.render("Покоління: " + str(current_generation), True, (0, 0,
0))
        text_rect = text.get_rect()
        text_rect.center = (900, 450)
        screen.blit(text, text_rect)

        text = alive_font.render("Залишилось: " + str(still_alive), True, (0, 0, 0))
        text_rect = text.get_rect()
        text_rect.center = (900, 490)
        screen.blit(text, text_rect)

        pygame.display.flip()
        clock.tick(60)  # 60 FPS


if __name__ == "__main__":
    # Load Config
    config_path = "./config.txt"
```

```
config = neat.config.Config(neat.DefaultGenome,
                            neat.DefaultReproduction,
                            neat.DefaultSpeciesSet,
                            neat.DefaultStagnation,
                            config_path)

# Create Population And Add Reporters
population = neat.Population(config)
population.add_reporter(neat.StdOutReporter(True))
stats = neat.StatisticsReporter()
population.add_reporter(stats)

# Run Simulation For A Maximum of 1000 Generations
population.run(run_simulation, 1000)
```

ДОДАТОК Б

```
[NEAT]
fitness_criterion     = max
fitness_threshold     = 100000000
pop_size              = 30
reset_on_extinction   = True

[DefaultGenome]
# node activation options
activation_default      = tanh
activation_mutate_rate  = 0.01
activation_options      = tanh

# node aggregation options
aggregation_default     = sum
aggregation_mutate_rate = 0.01
aggregation_options     = sum

# node bias options
bias_init_mean          = 0.0
bias_init_stdev         = 1.0
bias_max_value          = 30.0
bias_min_value          = -30.0
bias_mutate_power       = 0.5
bias_mutate_rate        = 0.7
bias_replace_rate       = 0.1

# genome compatibility options
compatibility_disjoint_coefficient = 1.0
compatibility_weight_coefficient   = 0.5

# connection add/remove rates
conn_add_prob           = 0.5
conn_delete_prob        = 0.5

# connection enable options
enabled_default         = True
enabled_mutate_rate     = 0.01

feed_forward            = True
initial_connection      = full

# node add/remove rates
```

```
node_add_prob           = 0.2
node_delete_prob        = 0.2

# network parameters
num_hidden              = 0
num_inputs              = 5
num_outputs             = 4

# node response options
response_init_mean      = 1.0
response_init_stdev     = 0.0
response_max_value      = 30.0
response_min_value      = -30.0
response_mutate_power   = 0.0
response_mutate_rate    = 0.0
response_replace_rate   = 0.0

# connection weight options
weight_init_mean        = 0.0
weight_init_stdev       = 1.0
weight_max_value        = 30
weight_min_value        = -30
weight_mutate_power     = 0.5
weight_mutate_rate      = 0.8
weight_replace_rate     = 0.1

[DefaultSpeciesSet]
compatibility_threshold = 2.0

[DefaultStagnation]
species_fitness_func = max
max_stagnation       = 20
species_elitism      = 2

[DefaultReproduction]
elitism              = 3
survival_threshold = 0.2
```