Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра прикладної математики
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Математичні моделі та методи навчання з підкріпленням для задачи автономного водіння автомобіля
(тема)

Виконав:
студент _2_ курсу, групи _САУм-19-1
Кравченко М.О.
(прізвище, ініціали)
Спеціальність
124 Системний аналіз
(код і повна назва спеціальності)
Тип програми _освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма
Системний аналіз та управління
(повна назва освітньої програми)
Керівник ____доц. Єсілевський В.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ПМ _____ ____Тевяшев А.Д._____
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет _інформаційно-аналітичних технологій та менеджменту_____
Кафедра _прикладної математики_____
Рівень вищої освіти _другий (магістерський)_____
Спеціальність _124 Системний аналіз_____
(код і повна назва)
Тип програми _освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _Системний аналіз і управління_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ПМ _____
(підпис)
“____” _____ 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Кравченко Микиті Олексійовичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи _Математичні моделі та методи навчання з підкріпленням для____
задачи автономного водіння автомобіля_____
_____
затверджена наказом по університету від 23 жовтня 2020 р. № 1420 Ст_____

2. Термін подання студентом роботи до екзаменаційної комісії __10 грудня 2020 р.

3. Вихідні дані до роботи __данні з фронтальної камери авто та сенсорів_____
_____
_____

4. Перелік питань, що потрібно опрацювати в роботі _____
1. Системний аналіз проблеми машинного навчання у невідомому зовнішньому середовищі_____
2. Вибір і обґрунтування методу розв’язання_____
3. Програмна реалізація_____
4. Результати обчислювального експерименту_____
5. Аналіз можливих застосувань_____
_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

1. Актуальність теми роботи
2. Постановка задачі
3. Системний аналіз проблеми
4. Метод чисельного аналізу
5. Результати обчислювального експерименту

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Підбір та вивчення технічної літератури за темою роботи	вересень 2020 р.	виконано
2	Вибір та обґрунтування методу	жовтень – листопад 2020 р.	виконано
3	Розробка алгоритму і програми	листопад – грудень 2020 р.	виконано
4	Проведення аналітичних досліджень та розрахунків	листопад – грудень 2020 р.	виконано
5	Робота над текстом пояснювальної записки	грудень 2020 р.	виконано
6	Представлення роботи на рецензію в ЕК	грудень 2020 р.	виконано

Дата видачі завдання _1_ вересня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Єсілевський В.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 69 с., 13 табл., 29 рис., 1 дод., 12 джерела.

ГЛИБОКЕ НАВЧАННЯ З ПІДКРІПЛЕННЯМ, СИСТЕМА ДОПОМОГИ ВОДІЮ, МАШИННЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, АВТОМАТИЗАЦІЯ, РОЗПОДІЛЕННЕ НАВЧАННЯ З ПІДКРІПЛЕННЯМ, ШОССЕ.

Хоча деякі системи допомоги водію були комерціалізовані для забезпечення безпеки та зручності водію, вони можуть застосовуватися для автономного водіння в обмеженій ситуації, таких як шосе. У цій дипломній роботі буде запропоновано агента-супервізор, який може покращити DSA(driver assistent system) використовуючи глибоке розподілене навчання з підкріпленням. Агент-супервізор навчається з використанням наскрізного підходу, який напряму відображає зображення з камери та дані LIDAR у плані дій. Тому що хороша навчена глибока мережа з підкріпленням може привести до дій запобігання аварій яка додана для запобігання небезпечних ситуацій. На додаток, випадки управляння на шосе це стохастична середа з притаманними їй випадковостями, а тому його навчання проводиться за допомогою алгоритму розподіленого навчання з підкріпленням, яке є спеціалізованим для стохастичної середи. Оптимальна дія для автономного водіння обирається через розподіл повернутих значень. На закінчення, запропонований алгоритм перевіряється на трасі симулятора, який реалізовано ML-агентами Unity.

ABSTRACT

Introductory note: 69 pages, 13 tables, 29 figures, 1 appendixes, 12 sources.

DEEP REINFORCEMENT LEARNING, DRIVER ASSISTANCE SYSTEM, MACHINE LEARNING, CONVOLUTIONAL NEURAL NETWORK, AUTOMATION, DISTRIBUTIONAL REINFORCEMENT LEARNING, HIGHWAY DRIVING.

Even though some of the driver assistant systems have been commercialized to provide safety and convenience to the driver, they can be applied for autonomous driving in limited situations such as highways. In this graduate work, will be propose a supervisor agent, which can enhance the driver assistant systems by using deep distributional reinforcement learning. The supervisor agent is trained using end-to-end approach which directly maps both a camera image and LIDAR data into action plan. Because the well-trained network of deep reinforcement learning can lead to unexpected actions, collision avoidance function is added to prevent dangerous situations. In addition, the highway driving case is a stochastic environment with inherent randomness and, thus, its training is performed through the distributional reinforcement learning algorithm, which is specialized for stochastic environment. The optimal action for autonomous driving is selected through the return value distribution. Finally, the proposed algorithm is verified through a highway driving simulator which is implemented by the Unity ML-agents.

ЗМІСТ

ВСТУП

Для підвищення безпеки та зручності проведено багато випробувань в автомобільній галузі. Завдяки цим дослідженням, декілька систем допомоги водію вже були використані як комерційні технології для транспортних засобів. Серед них такі проекти, як адаптивний круїз-контроль(ACC), автоматичне екстрене гальмування(AEB) та система дотримання полоси руху(LKS).

Проте це займає багато часу щоб система допомоги водію була інтегрована на реальні дороги, оскільки потрібно серії калібрування алгоритму та тестування. Тепер увага прикута до автономного управління, як одному з рішень проблем з дорожнім рухом з точки зору безпеки та ефективності. На відміну від системи допомоги водію, які мають відносно легким індивідуальні цілі, автономне водіння вирішує високого рівня у складній дорожній середі.

Мета цієї роботи, використати існуючи комерційні системи допомоги водію для надійного автономного водіння. У обмежених ситуаціях на шосе, автономне водіння може здійснюватися шляхом об’єднання допоміжних систем, такі як ACC та LKS. Але у складному управлінні на шосе, мають бути інтегровані системи допомоги водію для безпечного та ефективного керування автомобілем.

1 СИСТЕМНИЙ АНАЛІЗ ПРОБЛЕМИ МАШИННОГО НАВЧАННЯ У НЕВІДОМОМУ НАВКОЛИШНЬОМУ СЕРЕДОВИЩІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Системний аналіз проблеми машинного навчання у невідомому навколишньому середовищі

1.1.1 Вербальна модель системи

Однією з важливих тем автономного водіння, являє собою спосіб знати найкращу політику водіння, яка гарантує безпеку та комфорт водієві. Проблема розв’язання цієї задачі полягає в тому, що політика водіння має задовольняти стійкості незалежно від умов дорожнього руху. Тем не менше, правила основних алгоритмів не можуть впоратися з різними умовами.

В останній час, глибоке навчання, яке поєднує в собі глибоку нейромережеву модель та навчання з підкріпленням, показує значний прогрес у багатьох сферах. Алгоритми глибокого навчання з підкріпленням виконують самонавчання на основі різного досвіду та отримують хорошу швидкодію з використанням багатомірних вхідних даних, таких як необроблене зображення. Використовуючи ці переваги, ми підходимо до проблеми автономного водіння з глибоким навчанням з підкріпленням.

Наш агент знаходиться у деякому стані та обирає з множини дій якусь дію. В результаті цього агент переходить до іншого стану та отримує відгук середи. Відгуком середи на прийняті рішення є сигнали підкріплення, тобто, агент взаємодіє з середою у результаті чого отримує деяку винагороду.

В залежності від результату дії винагорода може бути як позитивною так и негативною. Алгоритм машинного навчання намагається максимізувати винагороду яку отримую агент і тим самим навчається.

1.1.2 Морфологічний опис системи

Об'єктом дослідження є процес автоматизації керування автомобілем.

Предметом дослідження є навчання агента за допомогою алгоритмів глибокого машинного навчання.

Морфологічний опис задачі машинного навчання з підкріпленням почнем з розгляду поняття навколишнього середовища.

Навколишнє середовище – формулюється як марковський процес прийняття рішень, і в цьому сенсі алгоритм навчання тісно зв’язаний з динамічним програмуванням.

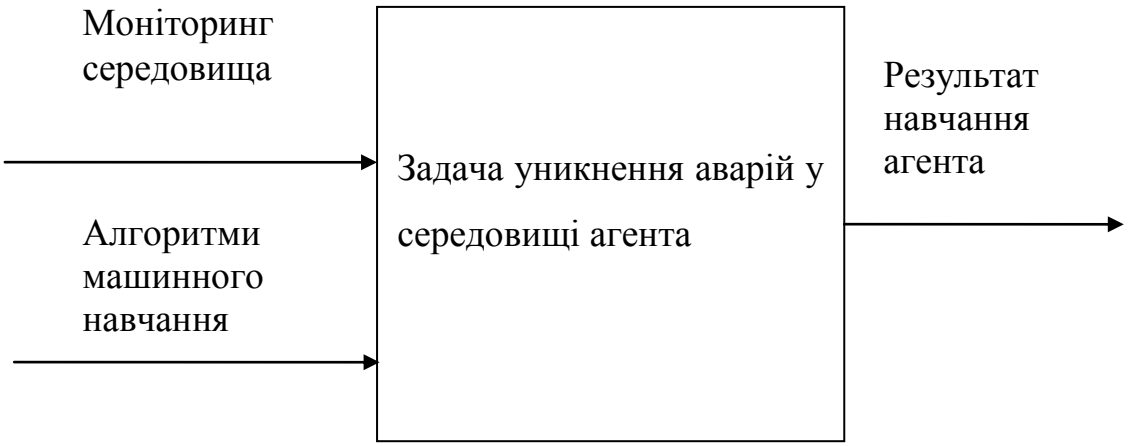

Рисунок 1.1 – Модель системи типу «чорний ящик»

Модель «чорний ящик» – модель досліджуваної системи, що зосереджена на досліджені реакції системи, як цілого, на зміни зовнішнього середовища, модель зображена на рисунку 1.1. До факторів моделі відносяться моніторинг середовища та алгоритми машинного навчання, кожен з яких має фіксовану кількість можливих значень, що називаються рівнями. Від зміни рівня одного із входів залежить вихідний стан «чорного ящика», тобто результат роботи такої моделі – навчання агента уникати аварії на дорозі.

1.1.3 Функціональна модель системи

Контекстна діаграма є вершиною деревовидної структури діаграм і являє собою саме загальний опис системи та її взаємодії з зовнішнім середовищем, вона зображена на рисунку 1.2.

Функціональна модель системи може бути представлена графічно за допомогою контекстної діаграми IDEF0. У такому випадку система постає у вигляді взаємодіючих функцій, або, інакше кажучи, функціональних блоків.

Спочатку проводиться загальний опис системи. Після опису системи в цілому проводиться розбиття її на великі фрагменти. Цей процес називається функціональної декомпозицією, а діаграми, які описують кожен фрагмент і взаємодія фрагментів, називаються діаграмами декомпозиції.

Після декомпозиції контекстної діаграми проводиться декомпозиція кожного великого фрагмента системи на більш дрібні і так далі, до досягнення потрібного рівня подробиці опису.

Після того, як контекст описаний, проводиться побудова наступних діаграм в ієрархії. Кожна наступна діаграма є більш докладним описом однієї з робіт на указаній вище діаграмі.

Рисунок 1.3 являє собою декомпозицію минулої діаграми на 4 функціональні блоки, між якими послідовні прямі зв’язки. Оскільки задача потребує детального пояснення, то цей блок підлягає декомпозиції, створивши тим самим наступний рівень декомпозиції із трьома функціональними блоками.

IDEF3 є стандартом документування технологічних процесів, що відбуваються на підприємстві, і надає інструментарій для наочного дослідження і моделювання їх сценаріїв. IDEF3 широко застосовується при розробці інформаційних систем. При цьому використовується інструмент візуального моделювання бізнес-процесів. Система описується як упорядкована послідовність подій з одночасним описом об'єктів, що мають відношення до процесу, що моделюється.

Розглянемо опис роботи у стандарті IDEF3 на рисунках 1.6 – 1.8.

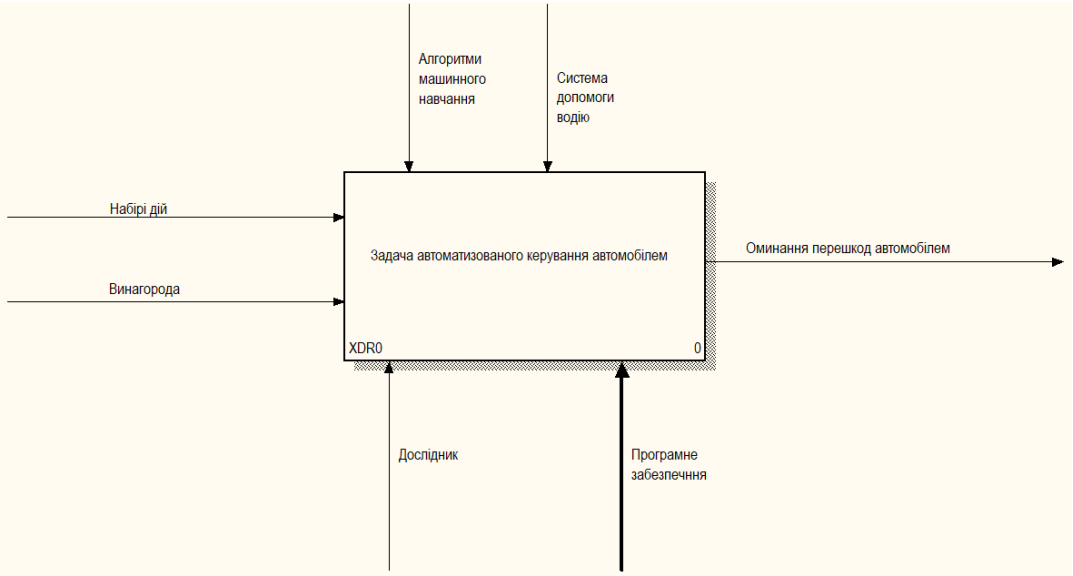

Рисунок 1.2 – Контекстна IDEF0 діаграма (рівень A-0)

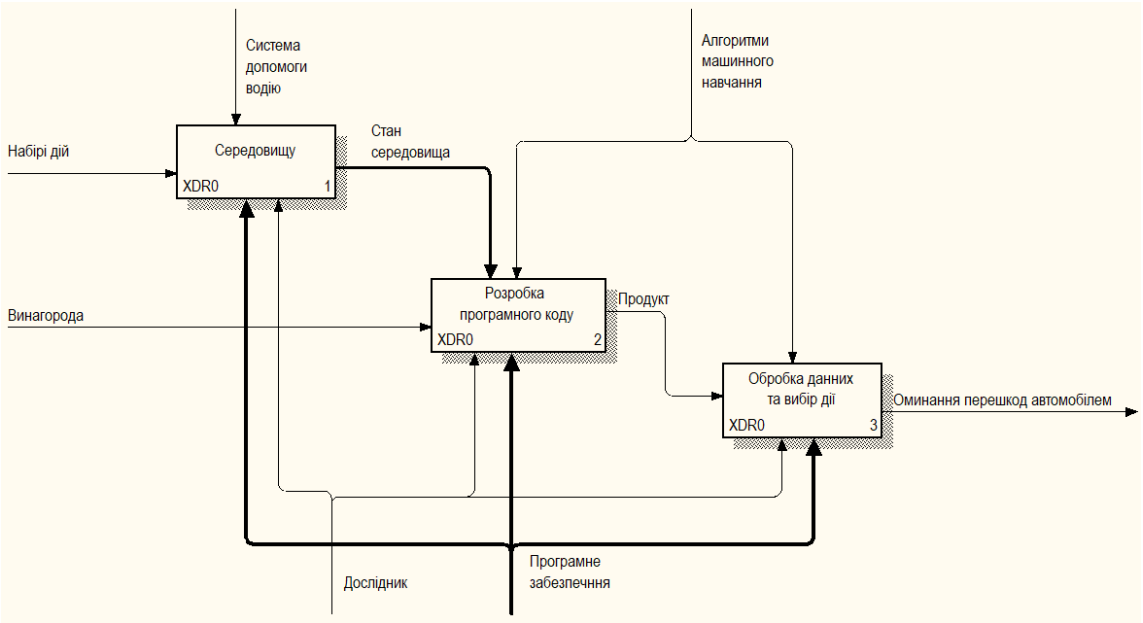

Рисунок 1.3 – Декомпозиція роботи «Навчання мережі за допомогою алгоритму навчання з підкріпленням» (рівень A0)

Розглянемо на рисунках 1.4 – 1.6 діаграми заданих декомпозицій.

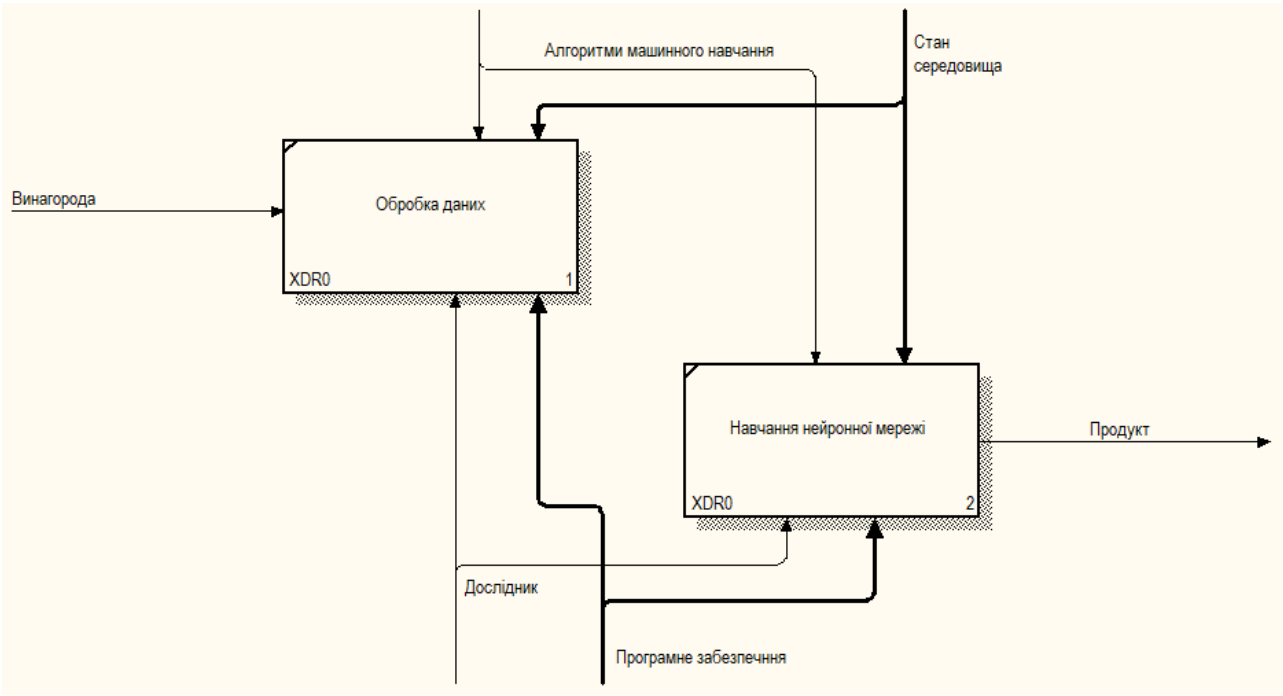

Рисунок 1.4 – Декомпозиція роботи «Написання програмного коду» (рівень А2)

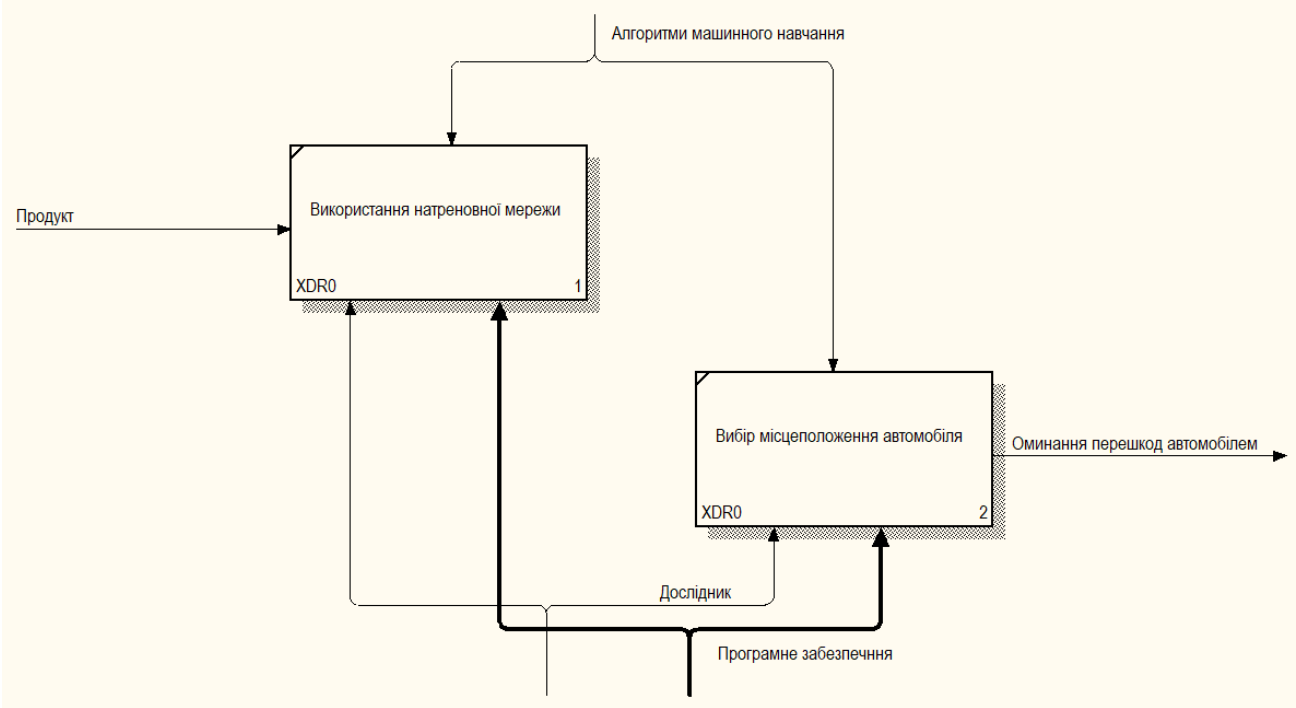

Рисунок 1.5 – Декомпозиція роботи «Створення автопілоту» (рівень А3)

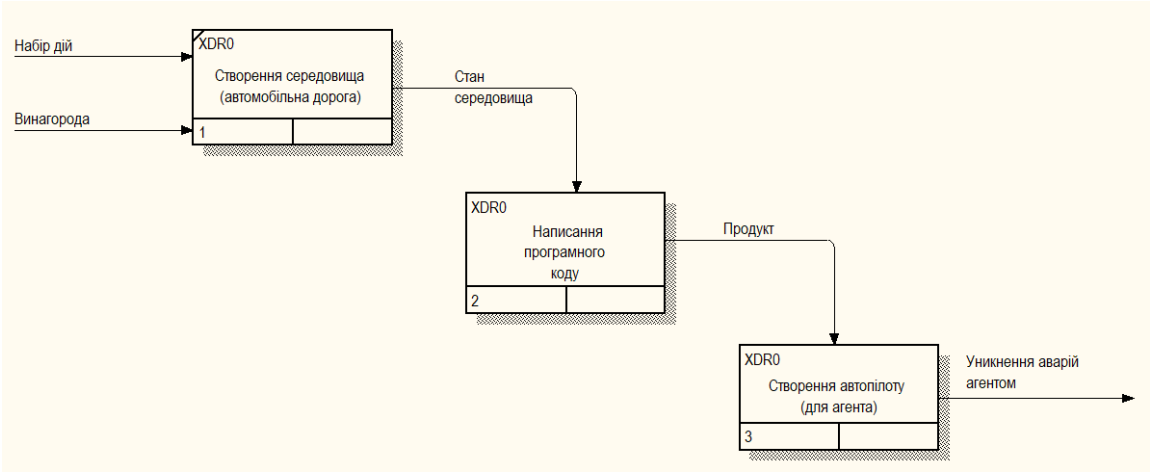

Рисунок 1.6 – Опис роботи «Навчання агента за допомогою алгоритму навчання з підкріпленням» (рівень A0 у нотації IDEF3)

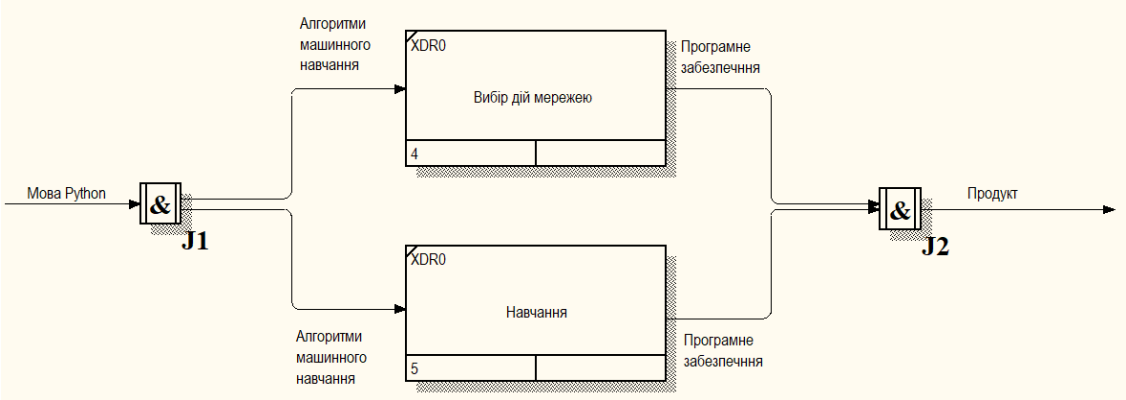

Рисунок 1.7 – Опис роботи «Написання програмного коду» (рівень A2 анотації IDEF3)

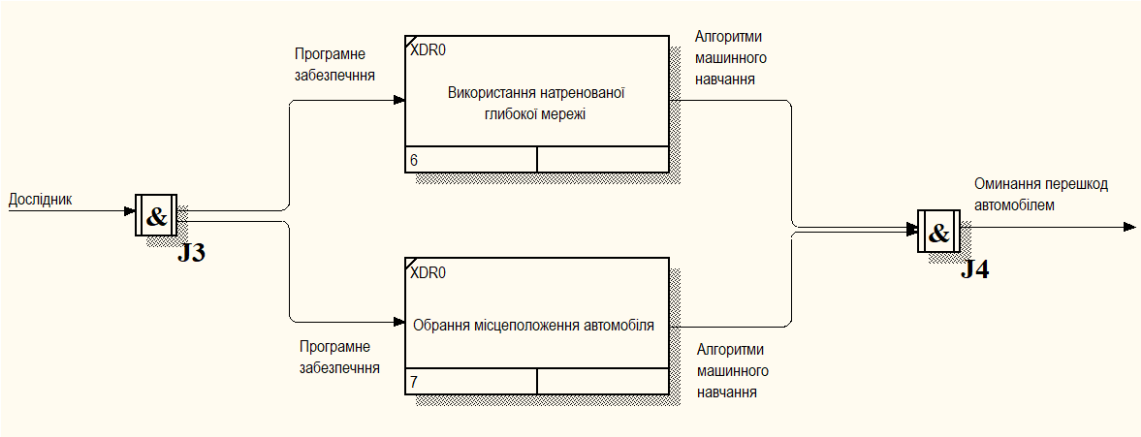

Рисунок 1.8 – Опис роботи «Створення автопілоту» (рівень A3 у нотації IDEF3)

1.1.4 Інформаційна модель

Інформаційні моделі відображають різні типи систем об'єктів, в яких реалізуються різні структури взаємодії і взаємозв'язку між елементами системи.

За допомогою інструмента методології DFD можна відображати джерела і адресати даних, ідентифікувати процеси і групи даних, що зв'язують в потоки одну функцію з іншого, і ефективно використовуються для опису процесів при впровадженні процесного підходу до управління організацією, так як дозволяє максимально знизити суб'єктивність опису бізнес процесів. Крім того, нотація DFD дозволяє описувати потоки документів і потоки ресурсів.

Ієрархію робіт в моделі показує діаграма дерева вузлів що дозволяє розглянути всю модель цілком, але не показує взаємозв'язку між роботами.

Діаграма дерева вузлів для всіх вузлів моделі показана на рисунку 1.9.

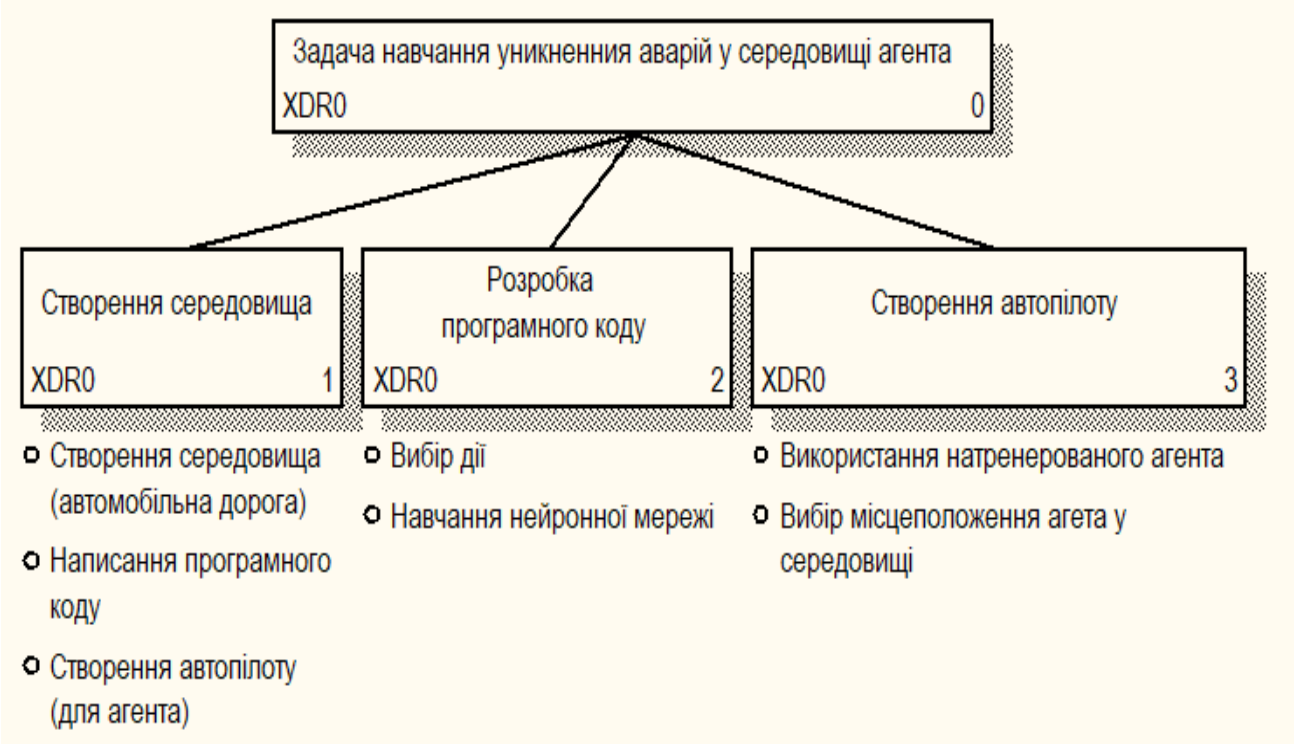

Рисунок 1.9 – Діаграма дерева вузлів

1.2 Аналіз сценаріїв вирішення проблеми машинного навчання у невідомому навколишньому середовищі

1.2.1 Модель аналізу проблеми

Перед тим, як проводити аналіз проблеми навчання з підкріпленням для створення автономного автомобіля, скористаємося методом аналізу ієрархій. У ролі критеріїв порівняння алгоритмів виступатимуть їх базові характеристики:

– швидкість навчання нейронної мережі (К1);

– стабільність (К2);

– складність алгоритму (К3);

– ефективність (К4).

Множина, з якої буде прийняте рішення, складається із наступних методів:

– метод DQN (А1);

– метод QR-DQN (А2);

– метод DDDQN (А3).

Таким чином, ієрархічна структура задачі складається із трьох рівнів:

– нульового – вибір методу роз’вязання задачі математичного моделювання динамічних об’єктів;

– першого – критерії порівняння методів;

– другого – множина альтернатив.

Далі побудуємо матрицю попарних порівнянь для кожного рівня ієрархії, скориставшись відповідною шкалою Т. Сааті. Ця шкала надає можливість поставити у відповідність ступеням переваги одного порівняльного об’єкта над іншим деяке число. На основі цих даних знаходяться вектори локальних пріоритетів, індекси узгодженості та відношення узгодженості.

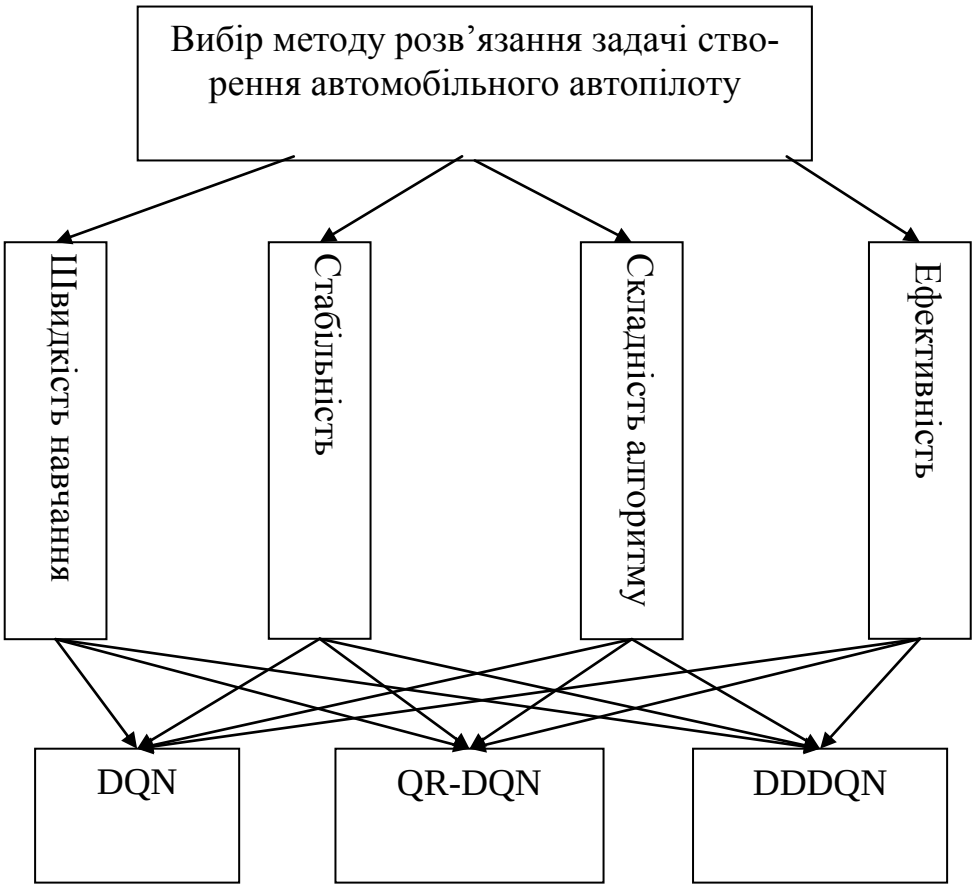

Рисунок 1.10 – Ієрархічна структура задачі вибору методу

Останні два допомагають визначити міру узгодженості результатів, отриманих експертним шляхом. Допустимим вважається значення відношення узгодженості $\leq 0,2$.

Індекс узгодженості (ІУ) $= \dfrac{4,07679-4}{4-1} = 0,02559$.

Відносна узгодженість (ВУ) $= \dfrac{0,02559}{0,9} = 0,02844$.

Вектор локальних пріоритетів відносно проблеми вибору набуває вигляду:

$$\vec{p}^{K} = \left(0,17757; 0,30231; 0,04698; 0,47314\right).$$

З отриманих проміжних результатів можна зробити висновок, що матри-

ця попарних порівнянь заповнена правильно, тобто жодне з тверджень не суперечить загальній логіці моделі, а дані узгоджені між собою на допустимому рівні.

Таблиця 1.1 – Матриця попарних порівнянь першого рівня

Номер п/п.	К1	К2	К3	К4	Власний вектор	Вектор приоритетів
К1	1	$\frac{1}{2}$	5	$\frac{1}{3}$	0,95544	0,17757
К2	2	1	7	$\frac{1}{2}$	1,62658	0,30231
К3	$\frac{1}{5}$	$\frac{1}{7}$	1	$\frac{1}{7}$	0,25276	0,04698
К4	3	2	7	1	2,54573	0,47314

Індекс узгодженості (ІУ) $= \frac{3,0092-3}{3-1} = 0,0046$.

Відносна узгодженість (ВУ) $= \frac{0,0046}{0,9} = 0,00793$.

Вектор локальних пріоритетів: $\vec{p}_1^{\,A} = (0,16342;\ 0,29696;\ 0,53962)$.

Аналогічним способом необхідно провести порівняльний аналіз альтернатив відносно кожного з критеріїв. Таким чином ми отримаємо дані, що задовольняють індексу узгодженості та відносній узгодженості. Матриці попарних порівнянь представлені у вигляді таблиць 1.2 – 1.6.

Таблиця 1.2 – Матриця попарних порівнянь першого критерію

К1	А1	А2	А3	Власний вектор	Вектор пріоритетів
А1	1	$\frac{1}{2}$	$\frac{1}{3}$	0,55032	0,16342
А2	2	1	$\frac{1}{2}$	1	0,29696
А3	3	2	1	1,81712	0,53962

Індекс узгодженості (ІУ) $= \frac{3,00369-3}{3-1} = 0,00185.$

Відносна узгодженість (ВУ) $= \frac{0,00185}{0,58} = 0,00319.$

Вектор локальних пріоритетів: $\vec{p}_2^{\,A} = (0,12202;\ 0,64833;\ 0,22965).$

Таблиця 1.3 – Матриця попарних порівнянь другого критерію

К2	А1	А2	А3	Власний вектор	Вектор пріоритетів
А1	1	$\frac{1}{5}$	2	0,46416	0,12202
А2	5	1	3	2,46621	0,64833
А3	2	$\frac{1}{3}$	1	0,87358	0,22965

Індекс узгодженості (ІУ) $= \frac{3,06489-3}{3-1} = 0,03245.$

Відносна узгодженість (ВУ) $= \frac{0,03245}{0,58} = 0,05594.$

Вектор локальних пріоритетів: $\vec{p}_3^{\,A} = (0{,}18839;\ 0{,}73065;\ 0{,}08096)$.

Таблиця 1.4 – Матриця попарних порівнянь третього критерію

К3	А1	А2	А3	Власний вектор	Вектор пріоритетів
А1	1	$\frac{1}{5}$	3	0,84343	0,18839
А2	5	1	7	3,27107	0,73065
А3	$\frac{1}{3}$	$\frac{1}{7}$	1	0,36246	0,08096

Індекс узгодженості (ІУ) $= \frac{3{,}18612 - 3}{3-1} = 0{,}09306$.

Відносна узгодженість (ВУ) $= \frac{0{,}09306}{0{,}58} = 0{,}16045$.

Вектор локальних пріоритетів: $\vec{p}_4^{\,A} = (0{,}15669;\ 0{,}32593;\ 0{,}51738)$.

Таблиця 1.5 – Матриця попарних порівнянь четвертого критерію

К4	А1	А2	А3	Власний вектор	Вектор пріоритетів
А1	1	$\frac{1}{2}$	$\frac{1}{3}$	0,55032	0,15669
А2	3	1	$\frac{1}{2}$	1,14471	0,32593
А3	3	2	1	1,81712	0,51738

Індекс узгодженості (ІУ) $= 0,07252$.

Відносна узгодженість (ВУ) $= \frac{0,07252}{0,9+0,58} = 0,049$.

Таблиця 1.6 – Кінцеві результати задачі вибору методу розв'язання

К / А	К1	К2	К3	К4	Вектор пріоритетів
А1	0,16342	0,12202	0,18839	0,15669	0,14889
А2	0,29696	0,64833	0,73065	0,32593	0,43726
А3	0,53962	0,22965	0,08096	0,51738	0,41384

На основі отриманих результатів розрахуємо вектор глобальних пріоритетів та відповідні показники узгодженості. Зазначимо, що спосіб розрахунку індексу узгодженості для всієї ієрархії відрізняється від попередніх випадків – це сума індексу для першого рівня ієрархії та скалярного добутку вектора локальних пріоритетів критеріїв з вектором індексів узгодженості відповідного критерію.

Як бачимо з таблиці 1.6 усі дані відповідають нормам узгодженості, а максимальна компонента вектора глобальних пріоритетів відповідає другій альтернативі, тобто спосіб, яким буде розв'язуватися задача, це QR-DQN метод.

1.2.2 Оцінювання вектора пріоритетів незадоволень методом аналізу ієрархій

Для продовження аналізу проблеми, зокрема оцінювання глобального вектора пріоритетів, необхідно побудувати матрицю попарних порівнянь першого рівня, а також і кожної виділеної властивості. Усі матричні дані наведені у відповідних таблицях 1.8 – 1.11.

Таблиця 1.7 – Матриця попарних порівнянь першого рівня

Номер п/п	Бажані властивості	Критичні властивості	Небажані властивості	Власний вектор	Вектор пріоритетів
Бажані властивості	1	5	3	2,466	0,631
Критичні властивості	$\frac{1}{5}$	1	$\frac{1}{4}$	0,368	0,094
Небажані властивості	$\frac{1}{3}$	4	1	1,075	0,275

Таблиця 1.8 – Матриця попарних порівнянь бажаних властивостей

Бажані властивості	Ефективність	Стабільність	Швидкість	Власний вектор	Вектор пріоритетів
Ефективність	1	2	3	1,817	0,539
Стабільність	$\frac{1}{2}$	1	2	1	0,3
Швидкість	$\frac{1}{3}$	$\frac{1}{2}$	1	0,55	0,161

Таблиця 1.9 – Матриця попарних порівнянь критичних властивостей

Критичні властивості	Складність	Багатофункці.	Власний вектор	Вектор пріоритетів
Складність	1	$\frac{1}{4}$	$0,5$	$0,2$
Багатофункц.	4	1	2	$0,8$

Таблиця 1.10 – Матриця попарних порівнянь небажаних властивостей

Небажані властивості	Границі	Похибка	Витрати	Власний вектор	Вектор пріоритетів
Границі	1	4	$\frac{1}{6}$	$0,322$	$0,075$
Схильність до перенавчання	$\frac{1}{4}$	1	$\frac{1}{5}$	$1,077$	$0,252$
Витрати	6	5	1	$2,885$	$0,673$

Упевнившись, що табличні дані не суперечать один одному (ІУ $= 0,069$, ВУ $= 0,06$), знайдемо значення вектора глобальних пріоритетів для кожної властивості і зобразимо їх у вигляді стовпчастої діаграми на рисунку 1.11.

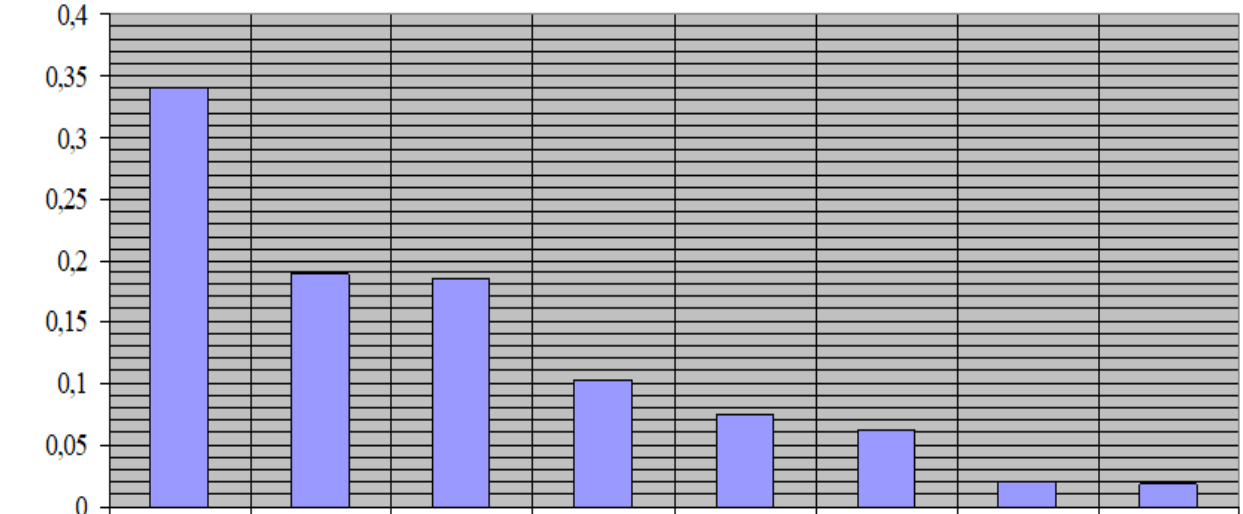

Рисунок 1.11 – Діаграма глобальних пріоритетів

На основі отриманих результатів зробимо висновок, що найвпливовішою трійкою властивостей є збільшення ефективності нейронної мережі, збільшення швидкості навчання та пристосування та збільшення стабільності. У зв'язку із цим зазначимо, що основна увага вирішення поставленої проблеми буде зосереджена на трійці властивостей.

Оскільки більша доля припадає на бажані властивості, то і зусилля відповідною мірою будуть направлені на покращення показників цих властивостей. Проте не слід забувати про наявність небажаної складової – певна частина ресурсів буде виділена на усунення великих обчислювальних витрат при моделюванні.

1.2.3 Модель вирішення проблеми

Розглянемо проблему якісного підвищення показників швидкодії при моделюванні середовища для навчання та навчання нейронної мережі.

Ієрархічна структура, що описує модель вирішення проблеми складається

із наступних рівнів:

– нульового – проблема, яку необхідно вирішити;

– першого – факторів, які найбільш впливають на проблему;

– другого – суб’єкти, діяльність яких найбільш впливова на проблему;

– третього – сценаріїв вирішення поставленої проблеми.

На основі даних глобального вектора пріоритету незадоволень, у рамках обраного методу були виділені наступні фактори впливу:

– схильність до перенавчання;

– границі кількості дій;

– витрати розрахунків.

Суб’єктами, від чиїх дій залежить результат вирішення проблеми, є:

– дослідник;

– керівник проекту;

– Міністерство освіти і науки;

У якості сценаріїв вирішення проблеми висунуті наступні альтернативи:

– математичний – алгоритмічна зміна існуючого та розробка нового методу моделювання;

– програмний – заміна технічного приладдя на більш вдосконалене та перехід на спеціалізоване ПО;

– науковий – підвищення кваліфікації дослідників та заохочення нових спеціалістів та роботи наукової галузі у досліджені автоматизації автомобіля.

На рисунку 1.12 зображена побудована ієрархічна структура, що повністю описує дану модель вирішення поставленої проблеми.

Оскільки більшість показників вже були знайдені, то на основі цих даних зробимо висновок, що найбільш прийнятним рішенням буде «математичний» сценарій, тобто вдосконалення вже існуючого методу або використання покращених версій методу, а саме методу QR-DQN.

Рисунок 1.12 – Ієрархічна структура вирішення проблеми

1.3 Змістовна та формальна постановка задачі

1.3.1 Змістовна постановка задачі

Навчання із підкріпленням (reinforcement learning) вивчає, як агент повинен діяти в середовищі, щоб максимізувати деякий довготривалий виграш. Алгоритми з частковим навчанням намагаються знайти стратегію, зробити так, щоб в кожному стані навколишнього середовища агент вибирав найкращу дію. В економіці і теорії ігор навчання з підкріпленням розглядається в якості інтер-

претації того, як може встановитися рівновага. При навчанні з підкріпленням, на відміну від навчання з учителем, не надаються вірні пари "вхідні дані-відповідь", а прийняття майже оптимальних рішень (що дають локальний екстремум) не обмежується явно. Навчання з підкріпленням намагається знайти компроміс між дослідженням невивчених областей і застосуванням наявних знань.

Формально найпростіша модель навчання з підкріпленням складається з:

– множини станів оточення S; A;

– множини дій A;

– множини "виграшів".

Загалом навчання із підкріпленням працює як на рисунку 1.13:

– на агента приходить стан;

– агент обирає дію;

– середовище посилає агенту нагороду і наступний стан;

– агент опрацьовує винагороду і коректує свої дії.

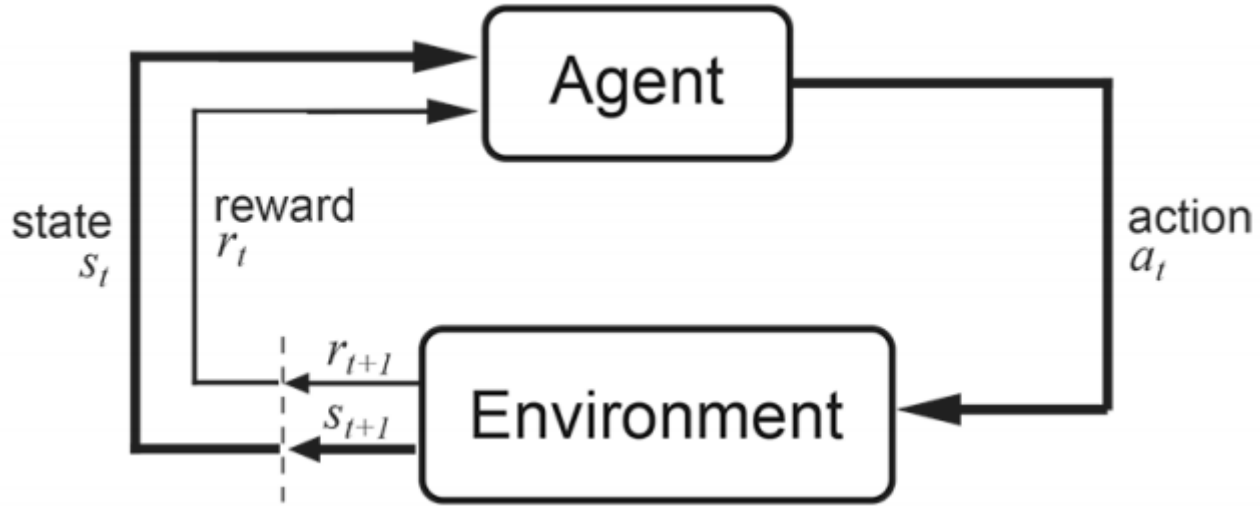

Рисунок 1.13 – Взаємодія агента та середовища

Розглянемо завдання створення автомобільного автопілоту шляхом навчання нейронної мережі за допомогою винагород. Як вже зазначалося, ключову роль в цьому процесі відіграє навчання з підкріпленням.

Як було зазначено вище навчання з підкріпленням намагається знайти компроміс між дослідженням невивчених областей і застосуванням наявних знань, тому першою задачею буде створення невідомої середи яку буде вивчати наша нейронна мережа.

Другим завданням є дослідження алгоритмів навчання з підкріпленням, ефективних для обмеженої множини дії агента.

Третьою, найголовнішим завданням, є вибір оптимальної стратегії з найбільш очікуваним виграшем, та найбільшою швидкістю навчання.

1.3.2 Формальна постановка задачі

Формальна постановка задачі навчання з підкріпленням має наступний вигляд.

Нехай S – множина станів середовища. Як бачимо на рисунку 1.13 гра агента зі середою проходить наступним чином. Спочатку визначається стратегія $\pi_1(a|S)$ та стан середовища S_1. Потім для усіх $t=1...T$ агент обирає дію $a_t : p_t(a|S_t)$, середовище генерує винагороду $r_{t+1} : p(r|a_t,S_t)$ та новий стан $S_{t+1} : p(S|a_t,S_t)$. І нарешті агент коректує стратегію $\pi_{t+1}(a|S)$. Це марковский процес прийняття рішень (МППР), якщо:

$$P(S_{t+1}=S', r_{t+1}=r \mid S_t, a_t, r_t, S_{t-1}, a_{t-1}, r_{t-1}, ..., S_1, a_1) =$$
$$= P(S_{t+1}=S', r_{t+1}=r \mid S_t, a_t).$$

Також слід зазначити що якщо $|A|<\infty$, $|S|<\infty$, то МППР називається фінітивним.

Тепер, коли була визначена функція виграшу, потрібно визначити стратегію, яка забезпечує найкращий результат.

1.4 Постановка задач дослідження

Існує два підходи у навчанні нейронних мереж для автопілоту: за допомогою симуляцій та на реальних даних. Також існує багато методів для навчання з підкріпленням.

Так як ми використовуємо саме навчання з підкріпленням або скорочено RL, то не слід забувати і про недоліки такого підходу та потрібно у ході дослідження мінімізувати негативний вплив. Потрібно зазначити: що, так як RL використовує функцію винагороди, то для правильної функція винагороди має охопити в точності те, що потребує задача. Це потрібно так як RL схильні до перенавчання, що веде до несподіваних послідовностей.

Дослідження безпечного автономного водіння, можна розділити на два підходи: перший – традиційне сприйняття, рамки планування та контролю, а другий методи основані на навчанні.

Методи, основані на навчанні, в останній час були популярні завдяки успіху процесів багатовимірних функцій, таких як згорткова нейронна мережа в області комп’ютерного зору [2–4]. Алгоритми навчання з підкріпленням можуть максимізувати очікувану суму винагород, та існує багато досліджень, щоб об’єднати ці дві технології для автономного водіння.

Для утримання полоси руху Rausch та інші [5] запропонували метод навчання мережі, яка напряму передбачає кут повороту на основі зображення, отриманого з фронтальної камери. Це вказує на те що мережа може визначити кут повороту для відстеження полоси шляхом автоматичного дослідження характеристик (полос руху і т.п.) з необробленого зображення фронтальної камери. John та інші [6] запропонували суміш експертних рамок для правильного розрахунку кута повороту для кожної сцени з використанням тривалої короткострокової пам’яті (LSTM). Кожний окремий експерт мережі, моделює поведінку досвідченого водія в певний момент даної дорожньої сцени, таке як прямий рух, поворот направо та наліво. Він добре працював на декількох послідовностях, так як вони розглядають різноманітні сцени водіння. Al-Qizwini та інші [7]

запропонували регресійну мережу, яка передбачає корисні стани для водіння, такі як помилки перетину, помилки напряму та відстань до перешкоди на зображені фронтальної камери, а не прогнозувати поворот керма прямо з зображення з передньої камери за допомогою структури GoogLeNet [8]. Кут повороту керма, дросельної заслінки та значення гальм розраховується з правила if-else базового алгоритму використовуючи цю інформацію.

Sallab та інші [9] запропонували метод навчання водіння, з політикою збереження полоси руху, з використання DQN (Deep Q Network) та DDAC (Deep Deterministic Actor Critic) з відсутніми перешкодами. Вони дослідили кермове управління, прискорення та уповільнення, щоб максимізувати очікувану винагороду на основі малорозмірних характеристики, таких як швидкість, місцеположення границі дороги.

У цій роботі за пропонується покращення структури водіння бакалаврської роботи з використання звичайного система допомоги водію (DAS) та глибокого навчання. Запропонований метод визначає функції DAS, такі як зміна полос руху, круїз-контроль та підтримку максимальної швидкості при пересуванні за полосою руху, а також обгін зі зміною декількох полос. Також буде доробле-на середа симуляції та додано нові перешкоди, такі як туман та шум сенсорів. Для перевірки запропонованого алгоритму, алгоритм має бути протестований при моделюванні ситуації з щільним рухом та продемонстровано, що продуктивність в процесі навчання при водінні покращується в умовах середньої швидкості, числа обгонів та кількості зміни полос руху.

2 ВИБІР ТА ОБҐРУНТУВАННЯ МЕТОДА РОЗВ'ЯЗАННЯ

2.1 Дослідження предметної області

Для початку потрібно ознайомитися з підходами для вирішення проблем, зв'язаних з навчанням з підкріпленням. Усього їх три, тому ми розглянемо усі.

Перший підхід — на основі значень. У навчанні з підкріпленням на основі значень метою є оптимізація функції $V(S)$. Функція value – це функція, яка інформує нас про максимальну очікувану винагороду, яку отримає система.

Значення кожної позиції — це загальна сума винагороди, яку система може тримати у майбутньому, починаючи з цієї позицій. Функцію можна розрахувати за формулою:

$$V_{\pi}(S) = E_{\pi}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | S_t = S \right], \quad (2.1)$$

де E_{π} – очікування;

R – винагорода;

S – поточний стан.

Система буде використовувати функцію значень (2.1) для вибору стану для кожного кроку. Вона приймає стан з найбільшим значенням.

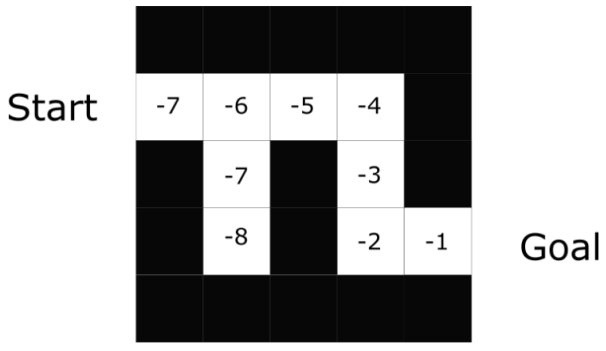

Рисунок 2.1 – Модель лабіринту для підходу на основі значень

На рисунку 2.1 зображений лабіринт, на кожному кроці будуть приймати-ся найбільші значення, тобто: -7, потім -6, -5 і так далі, щоб досягти мети.

Другий підхід – на основі політики. Цей підхід намагається напряму оптимізувати функцію політики $\pi(S)$ без використання функції значень.

Політика – це, те що визначає поведінку системи у даний момент часу, тобто $a = \pi(S)$. Це дозволяє нам зіставити кожну позицію з найкращою дією.

Існує два типи політики:

– детермінована: буде повертати одну і ту саму дію;

– стохастична: виводить ймовірність розподілу за діями.

Стохастична політика має вигляд:

$$\pi(a|S) = P\left[A_t = a | S_t = S\right]. \tag{2.2}$$

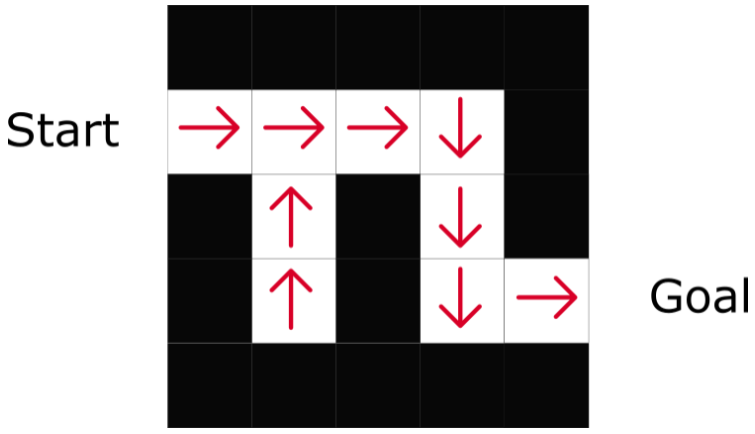

Рисунок 2.2 – Модель лабіринту для підходу на основі політики

Як можливо побачити на рисунку 2.2, політика показує на кращу дію для кожного кроку.

Третій підхід – на основі моделі. Цей підхід потребує створення моделі поведінки середи. Проблема кожної середи полягає в тому, що знадобляться різні уявлення даної моделі. Тому цей підхід використовується не дуже часто, і його розглядати не будемо.

2.2 Алгоритму навчання з підкріпленням

2.2.1 Q-learning та DQN

Reinforcement Learning зародився як доволі проста та оригінальна ідея, яка полягала у тому, щоб робити випадкову дію, а потім для кожної комірки у таблиці та кожного напряму руху, порахувати за спеціальною формулою, наскільки хороша ця комірка та обраний напрям. Чим більше отримане число, тим з більшою ймовірністю цей шлях спрямовує до перемоги.

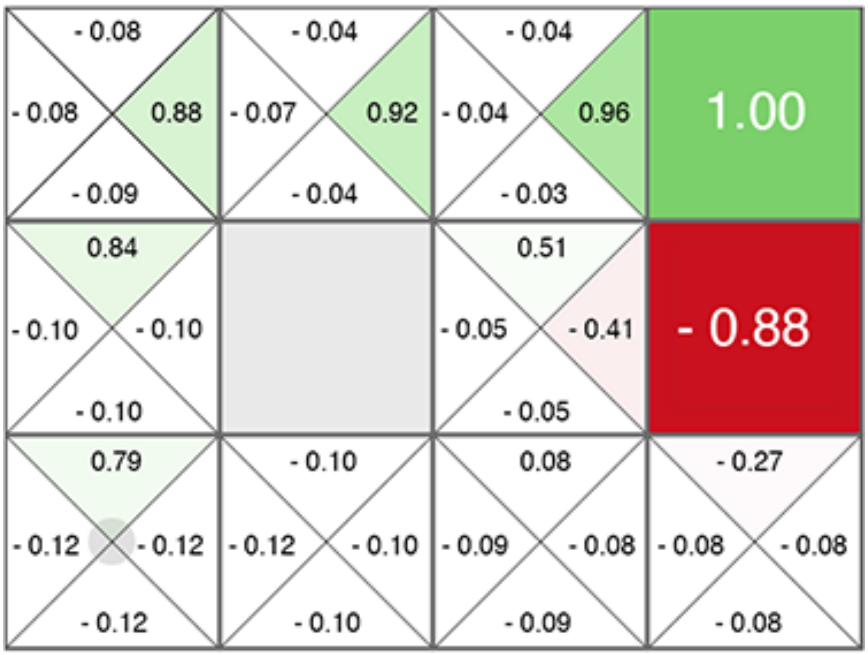

Рисунок 2.3 – Q-таблиця

У якій би комірці в таблиці на рисунку 2.3 ви не з’явились, потрібно рухатись по наростанню зеленого кольору, тобто у напряму максимального числа з боків поточної комірки.

Це число отримало назву Q (похідна від слова quality – якість вибору), а метод – Q-learning. Змінивши формулу розрахунку цього числа на нейронну мережу, а саме навчаючи нейронну мережу за цією формулою, у Deepmind отримали метод DQN [3], який у 2015 році переміг у багатьох Atari іграх та по-

клав початок революції у Deep Reinforcement Learning.

Нажаль, цей метод за своєю архітектурою працює тільки з дискретними діями. У DQN на вхід нейронної мережі подається поточний стан, а на вихід мережа передбачає число Q. А так як на виході мережа перераховані одразу усі можливі дії, то виходить що нейронна мережа у DQN реалізує класичну функцію $Q(S,a)$ з Q-learning, видає Q для state та action. Отже просто знаходять звичайний аргумент максимізації за масивом серед виходів мережі комірку з максимальним числом Q та робимо дію, яка відповідає індексу цієї комірки.

Слід зазначити, що можна завжди обирати дію з максимальним Q, тоді політика буде називатися детерміністською. Також можна обирати дію як випадкову з доступних, але пропорційно до їх Q-значень, тобто дія з високим Q буде частіше обиратися, ніж з низькою. Така політика називається стохастичною. Плюс стохастичного вибору полягає у тому, що автоматично реалізує пошук для дослідження світу, так як кожного разу обираються різні дії, які іноді не здаються найоптимальнішими, але які можуть у майбутньому привести до більшої винагороди. Тоді мережа навчиться та покращмо цією дією ймовірність, щоб тепер вони частіше обиралися згідно з ймовірністю.

Але що робити якщо варіантів дій нескінчена множина? Якщо це не 5 кнопок на контролері у Atari, а continuous момент керування двигуном робота? Звичайно, момент у діапазоні $-1...1$ можливо розбити на піддіапазони по 0.1, та в кожний момент часу обирати один з цих піддіпазонів, наче при натисканні контролера у Atari. Але не завжди потрібне число можна дискретизувати на інтервалі. Наприклад, якщо уявити що ми керуємо велосипедом на гірському піку. Можна повертати кермом лише на 10 градусів вліво або вправо. У якийсь момент часу пік може стати настільки вузькою, що поворот на 10 градусів в обох напрямках приведе до падіння. Це принципова проблема дискретних дій. А ще DQN не працює з великою розмірністю, та не працює навіть з 17 ступеня свободи просто не знаходиться. Але це маленький нюанс.

2.2.2 Double Deep Q Network (DDQN)

Double Deep Q Network – алгоритм, який був запропонований H.V. Hasselt та іншими [13].

У Double Deep Q Learning агент використовує дві нейронні мережі, щоб дізнаватися та передбачати, які дії виконувати на кожному етапі. Перша мережа називається, Q мережа або онлайн мережою, та використовуэться для прогнозування того, що треба зробити, коли агент зустрічається з новим станом. Вона отримує стан на вході та розраховує значення Q для можливих дій, які можуть бути вжиті. В агенті, який створено у цій роботі, онлайн мережа отримує вектор з шести значень(стани середи), у якості вхідних і вихідних векторів з трьох значень Q, одне для переміщення ліворуч, праворуч, та бездіяльність у теперешньому стані. Агент обирає дію яка має більше значення Q, яке розраховує онлайн мережа.

Double DQN вирішує проблему завищення Q-значень. Розраховуючи ціль за допомогою однієї глибокої мережі, ми стикаємось із простою проблемою: як ми можемо бути впевненими, що найкращою дією для наступного стану є дія з найбільшим Q?

Як відомо, точність значень Q залежить від того, яку дію ми спробували і які сусідні стани ми дослідили.

Отже, на початку дослідження мало інформації щодо кращого прийняття рішень. Тому, використання рішення з максимальним значенням Q(що є шумом), як найкращу дію може привести до неоптімальних дій. Якщо неоптімальним діям постійно присвоюється більш велике значення Q, ніж оптімально кращій дії, навчання буде важким. Рішення проблеми полягає у тому, щоб, коли ми вираховуємо значення Q, треба використовувати дві мережі, щоб розділити вибір дій від генерації цільового значення Q. Використовуємо наступні дії:

– використовуємо DQN мережу, щоб обрати, яку дію краще за усе обрати для наступного стану(дію з найбільшим значенням Q).

– використовувати цільову мережу, щоб використовувати цільове значен-

ня Q для виконнаня цієї дії у наступному стані.

Таким чином Double DQN допомагає зменьшити значення завищення значеня Q та допомагає тренувати швидше і мати більш стабільне навчання агента.

Цільове значення рівняння DQN та DDQN наступні:

$$y^{DQN} = R_t + \gamma \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}; \theta^-),$$

$$y^{DDQN} = R_t + \gamma Q(S_{t+1}, \arg\max_{A_{t+1}} Q(S_{t+1}, A_{t+1}; \theta^-); \theta^-).$$

Проаналізуємо за допомогою зображення дерева станів процес Double Q-Learning на рисунку 2.4.

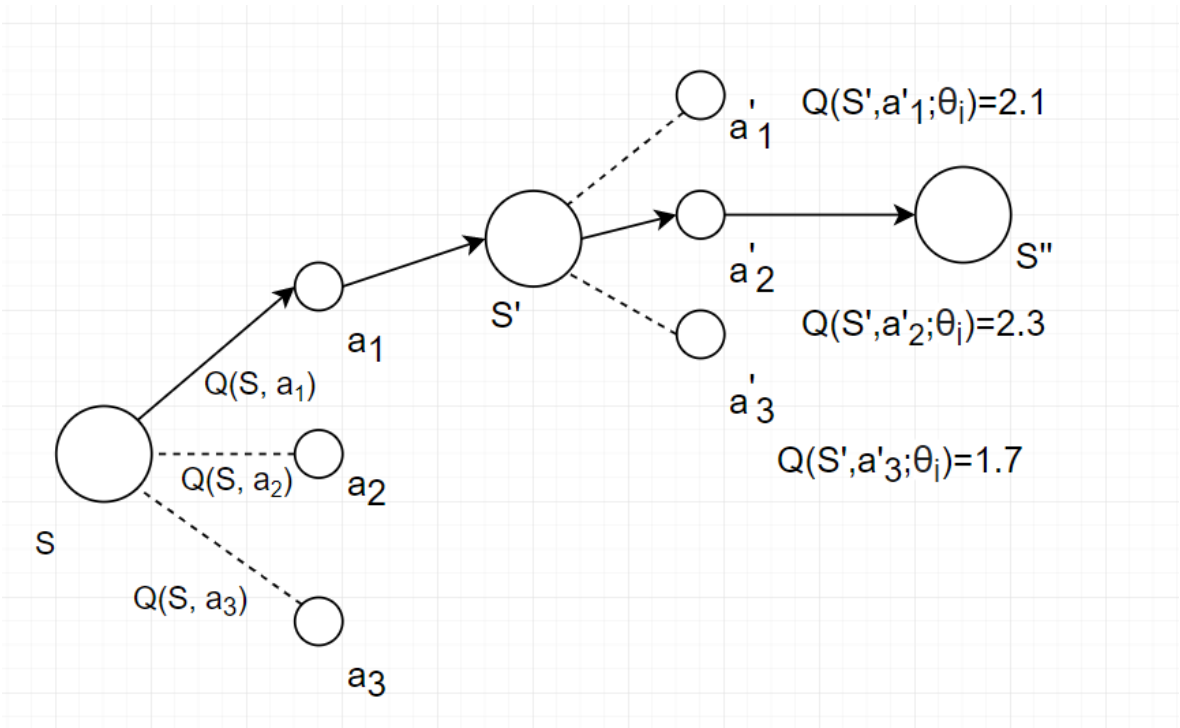

Рисунок 2.4 – Зображення дерева станів Double Q-Learning

AI агент знаходиться спочатку у стані s. Агент на основі деяких попередніх обчислень знає якості $Q(s,a_1), Q(s,a_2)$ та $Q(s,a_3)$ можливих двох дій у цих станах. Агент вирішує виконати дію a_1 та опиняється у стані s'.

Q-Network розраховує якості $Q(s', a_1')$, $Q(s', a_2')$ та $Q(s', a_3')$ можливих дій у цьому новому стані. Дію a_2' обрано бо вона має найбільше значення якості згідно Q-Network.

Нове значення дії $Q(s, a_1)$ для дії a_1 у стані s тепер може бути обчислено за допомогою схеми на наведеному вище рисунку, де $Q(s', a_1')$ це оцінка a_1', яка розраховується за допомогою Target-Network.

2.2.3 Dueling Deep Q Network (Dueling DQN)

Мережа Dueling Deep Q Network, це алгоритм, який має той же процес що й DQN. Однак, у неї є подвійна мережева архітектура, яка повністю зв'язує частину рівня DQN розділені на дві повністю зв'язані шари.

Перший шар оцінює значення стану, а другий шар оцінює переваги кожної дії. Після розрахунку повністю зв'язаного шару, значення Q отримується шляхом агрегування значення стану та переваг. Рівняння для розрахунку значення Q має наступний вигляд:

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + A(s_t, a_t; \theta, \alpha) - \frac{1}{|a|} \sum_{a_1} A(s_t, a_t; \theta, \alpha)),$$

де α – змінна функції переваг A;

β – змінна функції значення стану V;

$|a|$ – кількість дій.

Ця подвійна мережева архітектура отримує вражаюче зростання швидкодії порівнюючи з DQN, у багатьох іграх Atari. Також слід зазначити що Dueling DQN має стабільну швидкодію, навіть якщо збільшити кількість дій.

2.2.4 C51

C51 – один з розподілених алгоритмів RL, який був запропонований M.G. Bellemare та інші [7]. У цій статі вихідна мережа – це ймовірність розподілу. Значення осі X є опорами, і кожна опора $(\theta_i, i=\{1,...,N\})$ являє собою очікувану суму майбутньої винагороди. Значення розподілу по осі Y являє собою ймовірність $(q_i, i=\{1,...,N\})$ кожної опори, а N – кількість опор. Опори розподілені на однакові проміжки між мінімальними та максимальними значеннями $(V_{MIN}, V_{MAX}\})$. Таким чином опори фіксовані, а вихід мережі – це ймовірність опор за всіма діями. Тоді значення Q для кожної дії можна розрахувати наступним чином:

$$Q(o_{t+1}, a_{t+1}) = \sum_{i=1}^{N} q_i \theta_i (o_{t+1}, a_{t+1}). \tag{2.3}$$

Після того як значення Q будуть розраховані, дія буде обрана за допомогою ε-greedy. У цьому алгоритмі відтворення досвіду та техніка цільової мережі також використовується як і в DQN. Проте, як мережа оцінює розподіл як вихід, мета навчання також має бути отримана як розподіл, замість скалярного додатку. Таким чином, покращення Беллмана для кожного опори розраховується наступним чином:

$$\tau\theta_i = [r_t + \gamma\theta_i]_{V_{MIN}}^{V_{MAX}}, \tag{2.4}$$

де $\tau\theta_i$ – беллманівське покращення для $\{1,...,N\}$.

Після розрахунку беллманівського покращення для кожної опори, проекція $\tau\theta_i$ на опору θ_i має бути виконана. Після цього процесу, ймовірність цільового розподілу відносно кожної опори $\{1,...,N\}$ буде розраховано. Втрати для C51 – це дистанція між розподілами та втратами яку можна розрахувати за до-

помогою кроссентропії наступним шляхом:

$$-\sum_{i=1}^{N} m_i \log g_i(o_t, a_t).$$

2.2.5 QR-DQN

Розподільні алгоритми RL збігаються, коли вони задовольняють наступне γ - ущільнення.

$$\bar{d}_p(\tau^{\pi} Z_1, \tau^{\pi} Z_2) \leq \gamma \bar{d}_p(Z_1, Z_2),$$

де $\bar{d}_p$ – метрика по розподілу значень;

τ^{π} – розподіл оператора Беллмана;

Z_1, Z_2 – розподіл значення дій.

Щоб задовольнити γ - ущільнення, відстань Вассерштейна повинна бути метрикою відстані $\bar{d}_p$, а кроссентропія не задовольняє γ - ущільненню математично.

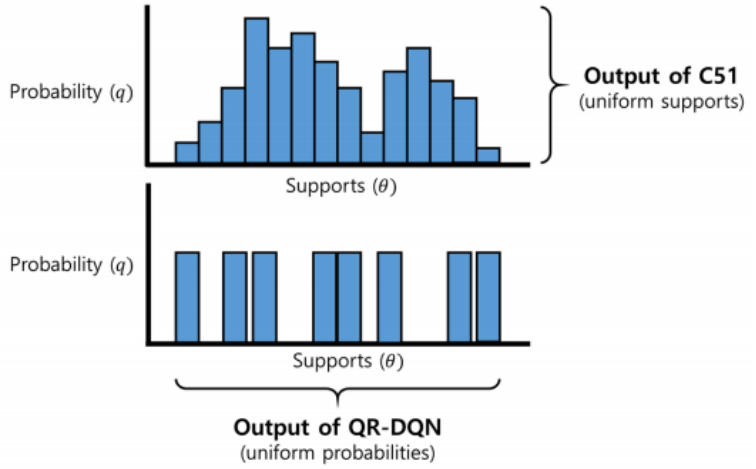

Рисунок 2.5 – Різниця між C51 та QR-DQN

Проте у C51 використовується кроссентропія як втрата для алгоритму, тому це алгоритм, який не доводить збіжність. Щоб вирішити цю проблему, W. Dabney та інші запропонував QR-DQN [10], який використовує квантильну регресію [14] для отримання опор з певною кількістю фіксованих, однорідних ймовірностей.

C51 – алгоритм, що оцінює ймовірності рівномірного розподілу опор, але QR-DQN – це алгоритм який оцінює опори рівномірно розміщених ймовірностей як показано на рисунку 2.5. Згідно з роботою [15], квантильна регресія техніки може зменшити відстань Вассерштейна між цільовим розподілом та прогнозуючим розподілом, так що це доводить що QR-DQN збігається математично.

Отримання значення Q та цільової опори аналогічні C51, наприклад рівняння 2.3–2.4, але велике кроки проектування C51 не потрібні. У цій роботі, квантильна втрата Хубера використовується для навчання, яке комбінує втрату квантильної регресії та втрату Хубера. Рівняння квантильної втрати Хубера описано наступним чином:

$$\sum_{i=1}^{N} E_j[\, p_{\tau_i}^{k}(\tau\theta_j - \theta_i(O_t, A_t))],$$

$$p_{\tau}^{k}(u) = \begin{cases} \dfrac{1}{2}u^2 \mid \tau - \delta_{\{u<0\}} \mid, & якщо \mid u \mid \le k, \\ k(\mid u \mid - \dfrac{1}{2}k \mid \tau - \delta_{\{u<0\}} \mid), \end{cases}$$

де θ це опори розподілу;

τ – центр однакових розподілів значення по осі ординат кумулятивної функції розподілу.

Тоді рівняння τ має наступний вигляд.

$$\tau_i = \frac{2(i-1)+1}{2N}, i = 1, \ldots, N,$$

де N – кількість опор.

QR-DQN доводить математичну збіжність як розподіл RL алгоритму порівнюючи з C51. Крім того, він показує себе краще по швидкодії порівнюючи з C51 в середі Atari. Більш того, цей алгоритм легко реалізується користувачем, тому що цей алгоритм не має процесу проектування та не потребує визначитися з діапазоном підтримки.

2.3 Вибір алгоритму для розв’язання поставленої задачи

У попередніх розділах ми ознайомилися з алгоритмами якими ми можемо використати для розв’язання задачі створення автопілоту. Тепер потрібно обрати алгоритм. Так як у магістерській роботі було прийнято рішення додати нові перешкоди для керування автомобілем, у вигляді туману, то також було вирішено використати більш потужні алгоритми ніж звичайний DQN. Я зупинився на двох з них, це DDQN та QR-DQN. Буде розглянуто саме ці два алгоритми та порівняно з класичним DQN.

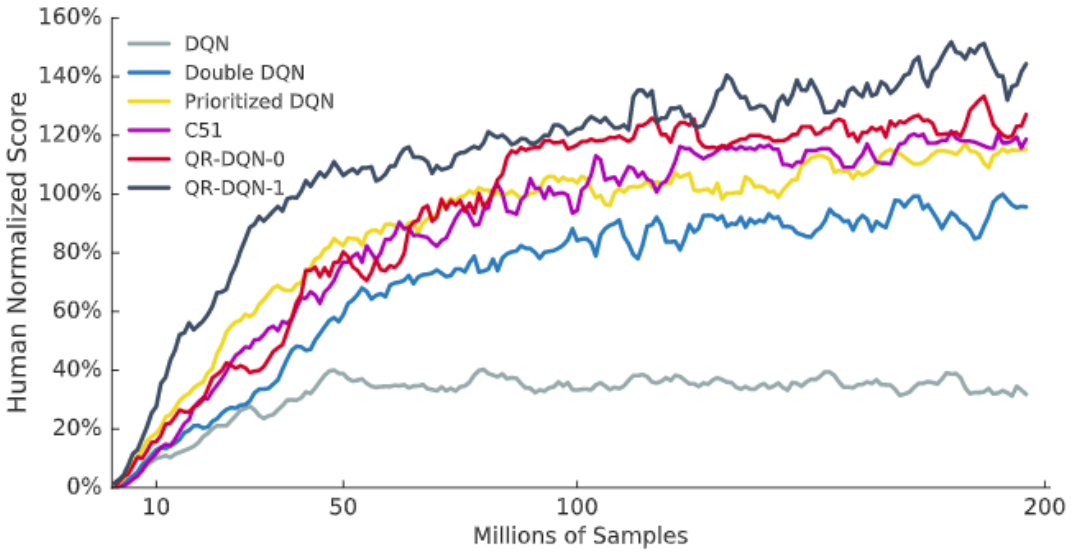

Рисунок 2.6 – Візуалізація даних отриманих у 2600 іграх Atari

	Mean	Median	>human	>DQN
DQN	228%	79%	24	0
DDQN	307%	118%	33	43
DUEL.	373%	151%	37	50
PRIOR.	434%	124%	39	48
PR. DUEL.	592%	172%	39	44
C51	701%	178%	40	50
QR-DQN-0	881%	199%	38	52
QR-DQN-1	**915%**	**211%**	**41**	**54**

Рисунок 2.7 – Порівняння DQN методів

Як бачимо на рисунках 2.6–2.7, QR-DQN з суворою квантільною втратою, тобто QR-DQN-0 (де k=0), та з квантильною витратою Хубера з k = 1, тобто QR-DQN-1 значно перевершують усі описані вище алгоритми та навіть C51.

Саме тому було вирішено реалізувати для навчання нейронної мережі DDDQN(DDQN + Dueling DQN) та QR-DQN замість C51 через його переваги які були описані вище. Також ми порівняємо дані отримані за допомогою нових алгоритмів та з реалізованим у бакалаврській роботі звичайним DQN алгоритмом.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Python та C# як мови високого рівня

Python – це потужний та гібридна мова програмування, яку можна використовувати для розробки web-додатків, для написання десктопних програм з графічним інтерфейсом.

C# – це простий, сучасний об’єктно-орієнтована і тіпобезопасна мова програмування, яка відноситься до широко відомого сімейства мов C.

Python має велику кількість безкоштовних бібліотек для розробки нейронних мереж, що робить цю мову незамінною при розробці програм з використанням нейронних мереж. Нейронна мережа у цій роботі буде написана з використанням саме цієї мови.

C# також має інструменти для створення нейронних мереж, наприклад ML.NET, або сервіс для машинного навчання Azure.ML. Але все ж вони поки що поступаються за функціоналом бібліотекам з Python. C# має дуже потужний інструмент для подібної задачі, він називається Unity.

Unity – це ігровий двигун, який дозволяє створювати ігри для багатьох платформ. Також Unity має власний сервіс для роботи з машинним навчанням, який називається ML-Agent. Він дозволяє написати нейронну мережу на мову Python, а потім сам використовує мережу у створеній грі.

CUDA-LIDAR – це проект який допомагає використовувати потік даних LIDAR, використовується для симуляції сенсорів авто.

Python було обрано з наступних причин.

1. Легкий поріг входження. Так як Python, динамічно типізована мова, це забезпечує легше розуміння та коротший код. Це дозволяє людині яка не знайома з цією мовою, дуже швидко вивчити її.

2. Простота розширення. Через те що, у стандартний дистрибутив Python вже входить багато бібліотек, не треба реалізувати деякі алгоритми та матема-

тичні функції власноруч, що також пришвидшує розробку.

3. Підтримка мультипарадігми. Python підтримує як об’єктноорієнтований, так і процедурний чи навіть функціональний стиль програмування. Це дозволяє обирати для різних алгоритмів навчання, саме ту парадигму яка їм підходить більше.

4. Універсальність та безкоштовність. Програми зроблені на Python, запускаються на усіх основних платформах, а також все це безкоштовно, і має велику підтримку зі сторони спільноти Python.

3.2 TensorFlow

TensorFlow (TF) [11] – доволі молодий фреймворк для глибокого машинного навчання, розроблений в Google Brain. Довгий час фреймворк розроблявся у закритому режимі під назвою DistBelief, але після глобального рефакторінгу 9 листопада 2015 року був опублікований у вільне користування.

Робота з TF будується навколо побудови графа розрахунків. Граф розрахунків – це конструкція, яка описує те, яким чином буду проведений розрахунок. У класичному імперативному програмуванні, код виконується порядково. У TF звичний імперативний підхід необхідний тільки для допоміжних цілей. Основа TF – це створення структури, яка задає порядок обчислень. Програми звичайним образом структурується на дві частини – створення графа розрахунків та виконання розрахунків у створеній структурі.

В TF граф складається з плейсхолдерів, змінних та операцій. З цих елементів можна зібрати граф, у якому будуть розраховуватися тензори.

Тензори – багатовимірні масив, вони слугують «паливом» для графу. Тензором може бути як окреме число, вектор ознак з розв’язуваної задачі чи зображення, так і цілий батч описів об’єктів чи масивів зображень. Замість одного об’єкта можна передати у граф масив об’єктів і для нього буде обчислений масив відповідей.

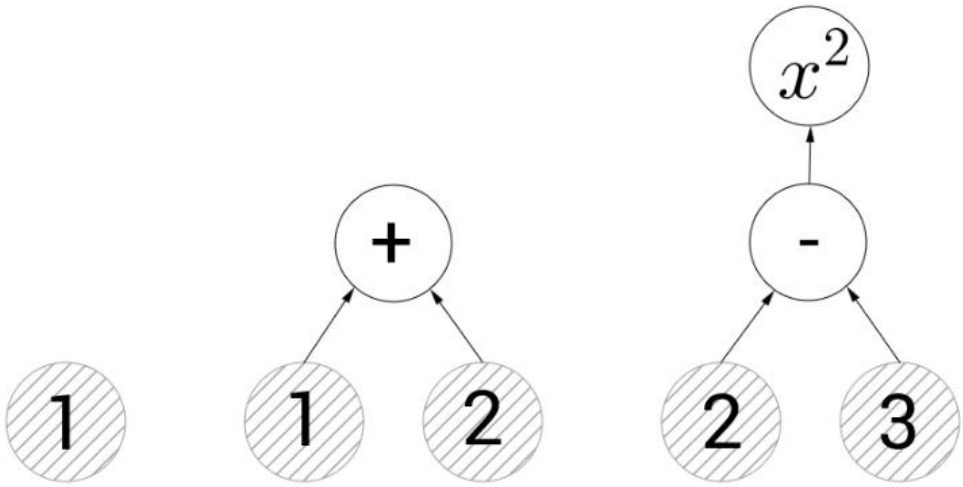

Рисунок 3.1 – Графи операцій у TF

Наприклад, на рисунку 3.1 зображено три графи. Перший граф зліва представляє константу із значенням 1. Другий граф, операція складання двох констант 1 та 2, що у результаті обчислення отримає відповідь 3. На третьому графі операція віднімання та зведення у квадрат, слід зазначити що спочатку буде проведена операція віднімання, а лише потім зведення у квадрат. Отже концепція графів розрахунку дуже проста.

3.3 Опис програми

3.3.1 Дії

Так як, програмі потрібні дані датчиків і зображення камери як вхідні дані, довелось знайти симулятор який забезпечує дані датчиків і зображення з камер. Крім того, транспортні засоби цього тренажера мають деякі функції безпеки. Ця функція застосовується до інших транспортних засобів та транспортного засобу версії ADAS (система допомоги водію). Функції безпеки:

– контролюється швидкість транспортного засобу, що дорівнює швидко ті транспортного засобу попереду;

– якщо відстань між двома транспортними засобами надто близька, швидкість знижується;

– немає зміни смуг;

– якщо транспортний засіб знаходиться в центрі смуги руху, переміщує автомобіль до центру смуги.

Простір дій для політики водіння визначений у дискретному просторі дій. Як ми використовуємо функції системи допомоги водієві, так і кожна дія для цієї системи, активує кожну з функцій DAS. У поздовжньому вигляді існує три види дій:

– круїз контроль зі швидкістю $\upsilon + \upsilon_{CC}$, де υ_{CC} це додаткова цільова швидкість що дорівнює 5 км/год.;

– круїз контроль зі швидкістю υ;

– круїз контроль зі швидкістю $\upsilon - \upsilon_{CC}$.

Очікується, що автономне екстрене гальмування та адаптивний круїз контроль, покриють ці поздовжні дії. Також маємо три види бокових дій:

– утримувати полосу руху;

– зміна полоси руху ліворуч;

– зміна полоси руху праворуч.

Так як автомобіль повинен рухатись як у поздовжньому так і в поперечному напрямі одночасно, можна визначити п’ять дискретних дій: прискорення, гальмування, зміна полоси руху ліворуч, зміна полоси руху праворуч, немає дій.

3.3.2 Функція винагороди

Якщо ми оберемо дію з точки зори навчання з підкріпленням, ми отримаємо винагороду в результаті дії. Як ми визначили раніше, проблема марковських випадкових процесів, має бути вирішена шляхом знаходження політики во-

діння, яка максимізує очікувану винагороду. Це означає, що оптимальна політика водіння може бути різною, в залежності від того, як визначені винагороди. Тому дуже важливо визначити винагороду так, щоб отримати правильну політику водіння.

Коли транспортний засіб рухається в умовах інтенсивного руху, він має задовольняти наступним трьом умовам:

– знайти політику, яка змушує транспортний засіб рухатися на великій швидкості;

– проїхати без зіткнень;

– не змінювати полосу руху занадто часто.

Тому були розроблені винагороди, щоб задовольнити ці умови.

$$r_{\upsilon}(\upsilon) = \frac{\upsilon - \upsilon_{\min}}{\upsilon_{\max} - \upsilon_{\min}} r_{\upsilon}, \max, \tag{3.1}$$

$$r_{col} = \begin{cases} -r_{collision} \\ 0 \end{cases}, \tag{3.2}$$

$$r_{lc} = \begin{cases} -r_{lanechange} \\ 0 \end{cases}, \tag{3.3}$$

$$r_{overtake} = \begin{cases} r_{overtake} \\ 0 \end{cases}, \tag{3.4}$$

$$r_{tot}(\upsilon) = r_{v}(\upsilon) + r_{col} + r_{lc} + r_{overtake}, \tag{3.5}$$

де υ – поточна швидкість транспортного засоба;

$\upsilon_{\max}, \upsilon_{\min}$ – максимальна та мінімальна швидкість за дорожніми правилами;

$r_{\upsilon,\max}$ – винагорода за максимальну швидкість (3.1);

υ_{lc} – штраф за зміну полоси руху (3.3);

$\upsilon_{collision}$ – штраф за зіткнення (3.2);

$r_{overtake}$ – винагорода за обгін (3.4).

З точки зору безпеки, зіткнення має бути запобігнене в умовах водіння. Саме тому зіткненню встановлюється найбільше значення, ніж усім іншим винагородам, щоб запобігти конфліктну поведінку в першу чергу. Так як, бажано мандрувати при $\upsilon_{\max}$, можна отримувати лінійну винагороду у відповідності з різницею між поточною та мінімальною швидкістю.

Оптимальна політика водіння змушує автомобіль переганяти автомобілі які повільно рухаються. Таким чином, обгони додаються до остаточної винагороди, щоб заохочувати пошук оптимальної політики.

Нарешті, так як небажано занадто часто змінювати смуг руху, значення υ_{lc} встановлене, щоб навчити їздити без зайвих змін смуг руху.

Таблиці 3.1 – Параметри винагороди

$r_{v,\max}$	1
$\upsilon_{\max}$	80 км/год.
$\upsilon_{\min}$	40 км/год.
$\upsilon_{collision}$	-10
υ_{lc}	-0.25
$r_{overtake}$	0.5

Усі параметри винагороди вказані у таблиці 3.1.

Таким чином намагаючись максимізувати виграш (3.5) за алгоритмом DQN, мережа намагається уникати зіткнень. Також слід зазначити що після навчання автомобіль під керуванням агент тримався на дорозі набагато впевненіше, і зводив до мінімуму зміну смуг.

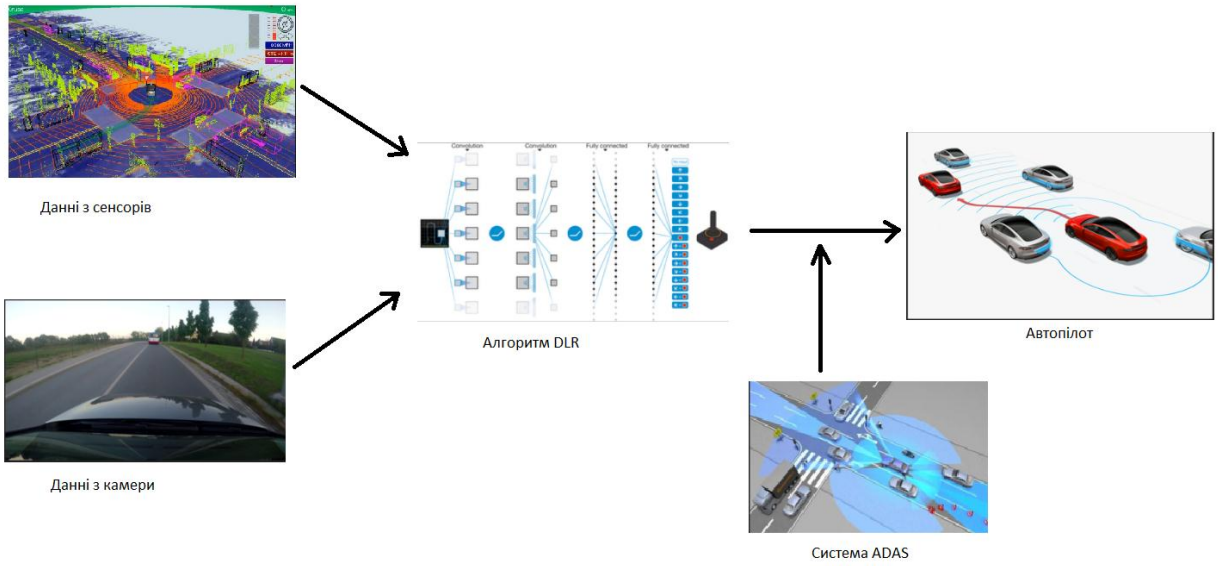

Рисунок 3.2 – Модель роботи програми

Загальна модель роботи програми наведена на рисунку 3.2. Як бачимо наш алгоритм DLR, а саме DQN, отримує данні з сенсорів та камери, обробляє їх та обирає потрібну дію. Також використовується система контролю безпеки водія ADAS як було описано вище. У кінці ми отримуємо автоматизований автомобіль.

3.4 Симулятор

Симулятор, який я обрав для написання цієї роботи, збудований з використання Unity та Unity ML-Agents. Змодельована дорожня середа – шосе, що складається з п’яти смуг руху.

Слід зазначити що транспортні засоби генеруються у центрі випадкової смуги руху у радіусі певної відстані від агенту. Крім агенту, алгоритм уникнення аварій застосовується для кожного транспортного засобу, тому що він запобігає зіткнення з іншими транспортними засобами. Якщо агент зіштовхується з іншим транспортним засобом або агентами, проїжджаючими на певній дистан-

ції, епізод закінчується, та середа обнуляється до первинного стану. Інші транспортні засоби випадковим чином виконують такі дії, як зміна смуг руху, прискорення, уповільнення за певний часовий проміжок.

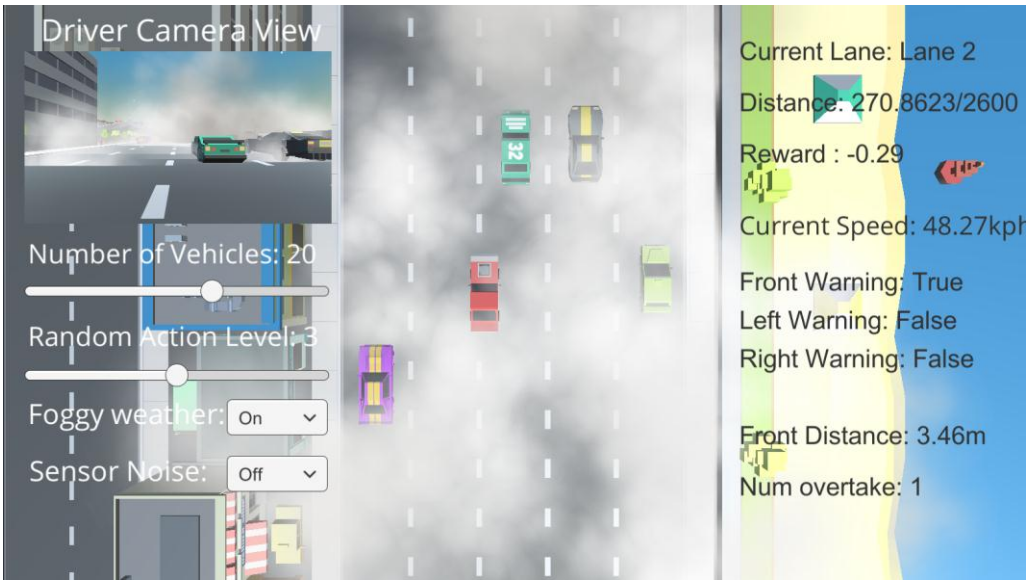

Рисунок 3.3 – Зразок роботи програми з туманом

Крім того у цій роботі було додано декілька факторів, які збільшують випадковість середи. Перший фактор це туман, наочно можна побачити його на рисунку 3.3. Туман погіршує огляд передньої камери. Він не тільки збільшує складність середи, але й збільшує випадковість. Другий фактор це шум сенсора, який додається до усіх датчиків LIDAR, для того щоб збільшити випадковість середи. До шуму сенсора додається наступне рівняння.

$$d_i = d_i + \alpha U(-d_i, d_i),$$

де i – індекс даних LIDAR;

d – відстань даних кожного сенсора; α – ваговий коефіцієнт шуму, який дорівнює 0.3;

$U(-d_i, d_i)$ – випадкова функція рівномірного розподілу яка довільно малює вибірку між $-d_i$ та d_i.

Алгоритми DRL навчаються за допомогою звичайного симулятору станів, при якому туман та шум датчиків не використовується. Після тренування, туман та шум датчиків додані при тестуванні DRL алгоритмів для перевірки роботи агента у стохастичній навколишній середі.

Різні дії інших транспортних засобів змінюють стан модельованої середи у багатьох аспектах, тому агент отримує багато досвіду у різних ситуаціях.

Спостереження за створеною середою у симуляторі реалізується двома методами: перший це масив діапазонів даних з сенсорів на зразок LIDAR, а другий візуальний. Оскільки у агента є камера на передній панелі, то на кожному кроці не оброблене піксельне зображення аналізується. Датчики мають один промінь на градус, та отримують данні в діапазоні 360 градусів. Якщо на об’єкт попадає промінь, він розраховує відстань між нашим агентом та об’єктом. Якщо перешкоди немає, то буде повернена максимальну відстань сенсорного зчитування для кожного кроку.

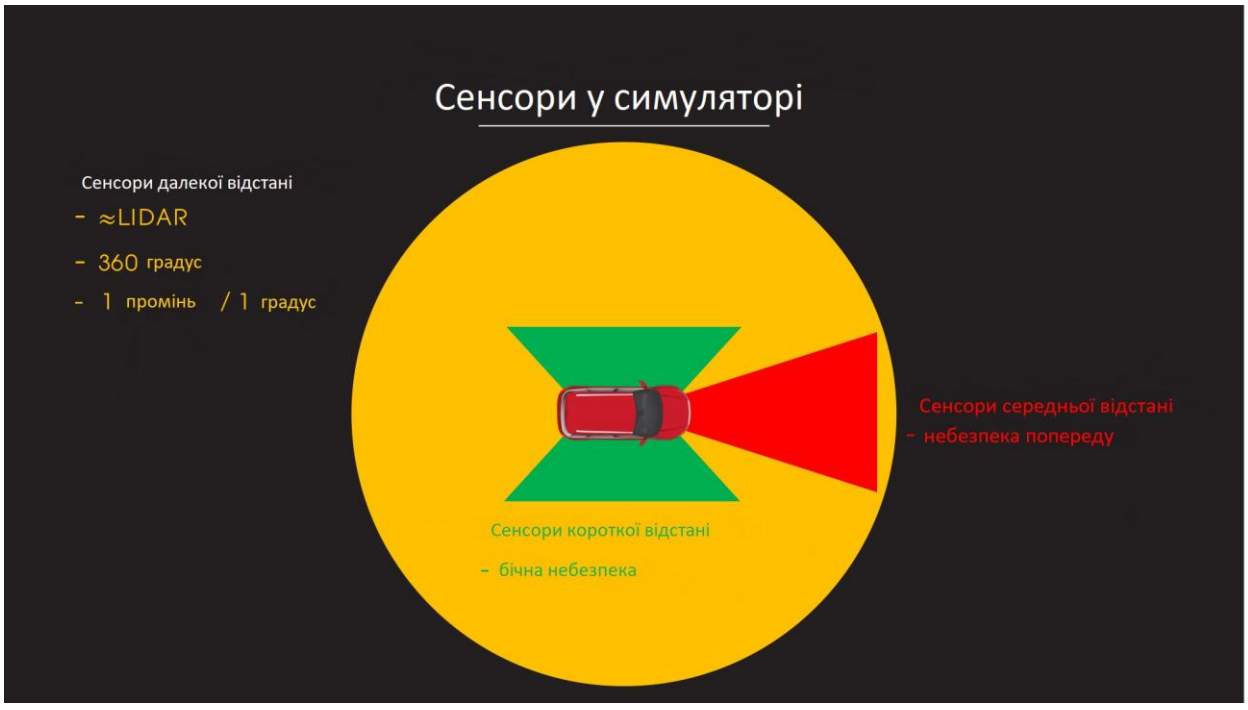

Рисунок 3.4 – Схема сенсорів у симуляторі

На рисунку 3.4 наглядно зображена схема сенсорів у симуляторі.

3.5 Архітектура мережі та гіпер-параметри

Для навчання керуванню, згорткові прошарки похідної структури DQN змінена для аналізу множинного введення, оскільки у запропонованому алгоритмі використовуються два різні входи одночасно.

У разі зображення з камери, данні обробляються за допомогою загорткової нейронної мережі, такої як DQN та інші, а данні з сенсорів обробляються за допомогою довгострокової пам’яті (LSTM). Після об’єднання цих двох оброблених даних, вони використовуються як вхід повністю зв’язаного прошарку, для отримання значень Q, які визначають дію. Алгоритм глибокого навчання має структуру як показано на рисунку 3.5.

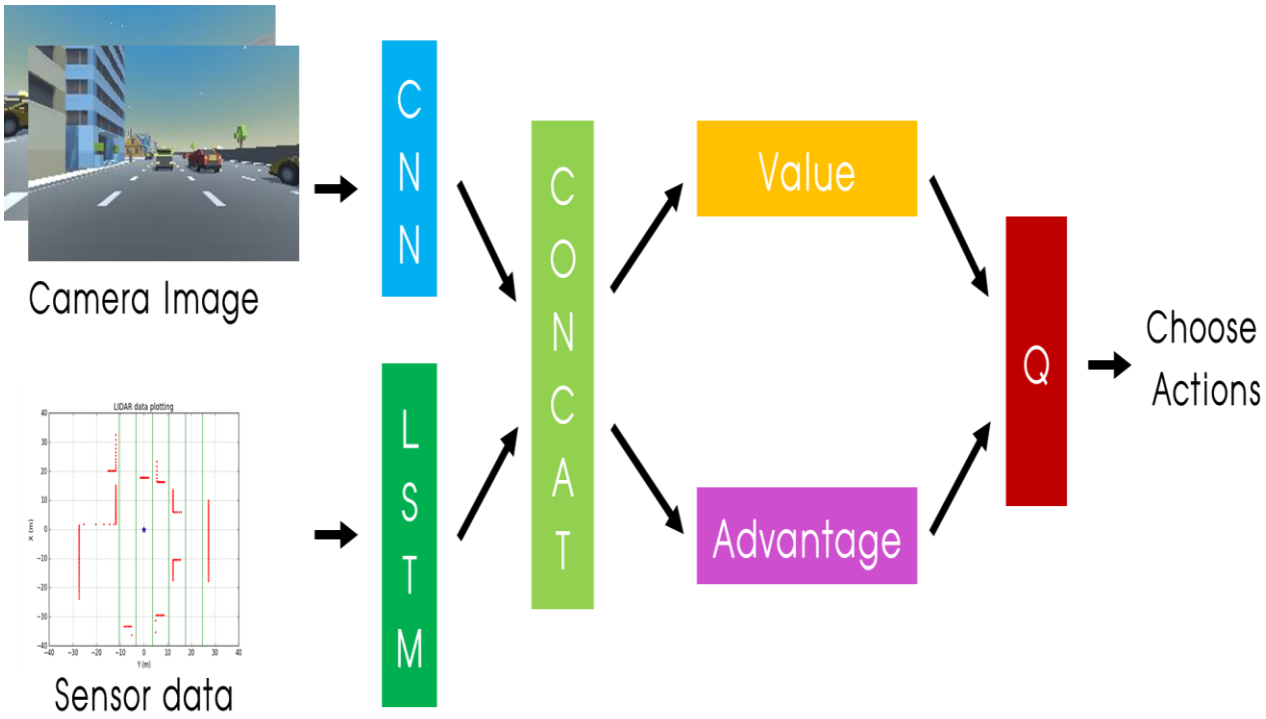

Рисунок 3.5 – Схема алгоритм глибокого навчання

Мережа навчалась за допомогою оптимізатора Adam [26] з epsilon 0.01/32, зі швидкістю навчання 0.00025 та міні-пакети розміром 32. Ваги мережі були ініціалізовані за допомогою Xavier ініціалізатора [27], та всі входи датчиків були нормалізовані у [-1;1]. Гіперпараметри CNN, LSTM та повнозв’язний шар

наведено у таблиці 3.2. Гіперпараметр частини CNN така сама, як і структура оригінального DQN, який широко використовується для інших алгоритмів DRL. Кількість опор θ встановлено на 200.

Розмір пам’яті відтворення було 100000, а коефіцієнт дисконтування γ був встановлений на 0.99. Було використано політику ε-greedy тривалістю 1000000, де було лінійно зменшено з 1.0 до 0.1 для кожного кроку, а потім зафіксовано на 0.1.

Таблиця 3.2 – Гіпер-параметри для політики водіння мережі

Данні	Тип	Функція активації	Гіпер-параметри
Данні з камери	Згортка	ReLU	patch size = (8*8), stride = 4 # of filters = 32
	Згортка	ReLU	Patch size = (8*8) stride = 4 # of filters = 32
	Згортка	ReLU	Patch size = (8*8) stride = 4 # of filters = 32
Сенсорні данні	LSTM	–	time steps = 4 # of cell states = 256
Об’єднані данні	Об’єднання	ReLU	# of units = 512

4 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ

Швидкодія розподіленого алгоритму RL перевірено в моделюванні шляхом порівняння з DQN та DDDQN(Double + Dueling) які не використовують розподіл. Крім того, швидкодія структури з множиною входів перевірено шляхом порівняння трьох результатів з різноманітними конфігураціями:

– використання тільки зображення з камери у вигляді вхідних даних;

– використання тільки даних LIDAR як вхідних;

– використання зображення з камери разом з даними LIDAR як вхідних.

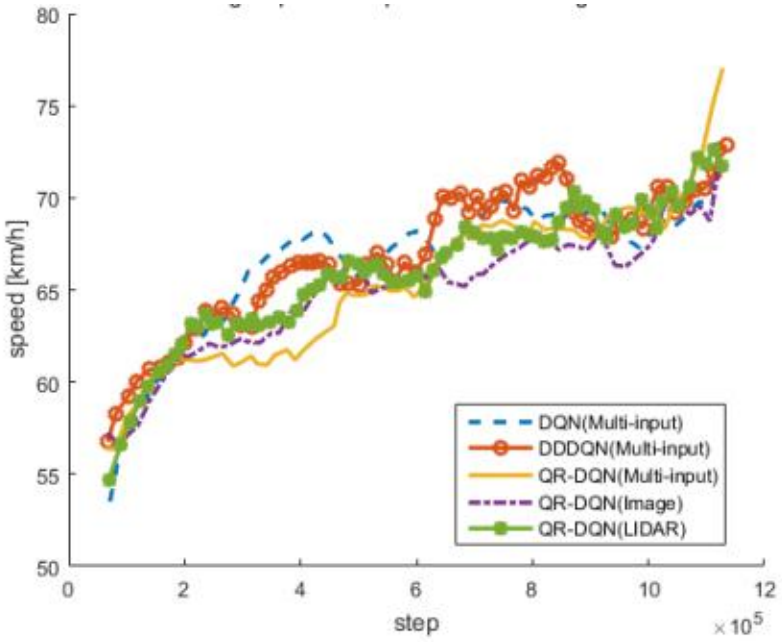

Рисунок 4.1 – Середня швидкість за 5 епізодів

На рисунку 4.1-4.2 продемонстрований процес навчання у симуляції. Навчання проводиться у нормальних умовах без туману та шуму на сенсорах. Ось X обох рисунків – кроки навчання у симуляції. На рисунку 4.1 зображена середня швидкість агента за 5 епізодів моделювання. На рисунку 4.1 можна побачити тенденцію зростання швидкості агента в процесі навчання. На рисунку 4.2 наведено середнє значення зміни смуг руху для агента за 5 епізодів симуляції. На цьому рисунку можна побачити тенденцію зменшення кількості змін смуг руху для агента в процесі того як навчання прогресує. Наведений вище резуль-

тат доводить що навчання було успішно виконано згідно з встановленій функції винагород.

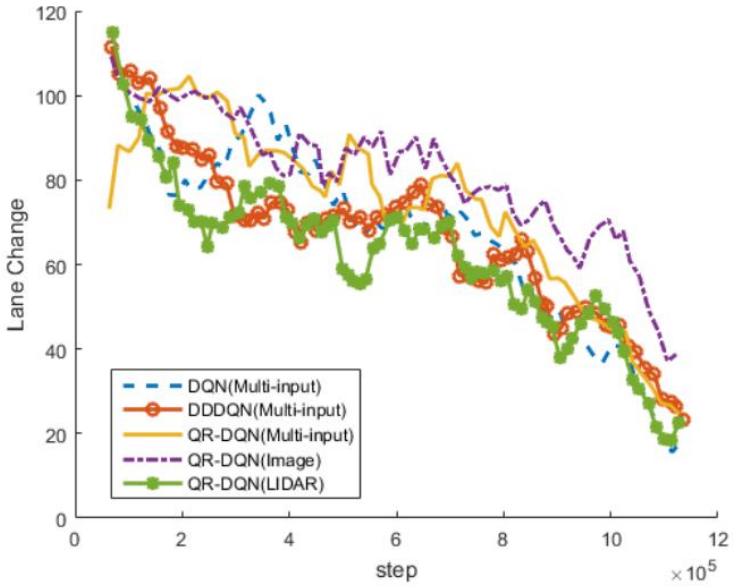

Рисунок 4.2 – Середня кількість зміни смуг руху за 5 епізодів

Таблиця 4.1 – Данні отримані з проведених експериментів

Вхідні данні	DQN (Multi)	DDDQN (Multi)	QRDQN (Image)	QRDQN (LIDAR)	QRDQN (Multi)
Швидкість (км/год)	66.10	67.78	60.92	69.06	72.38
σ – Швидкість	1.93	1.43	1.79	1.52	1.72
Зміна полос	42.25	26.31	13.18	44.28	15.98
σ –Зміна полос	5.50	2.58	1.63	6.09	1.61

На таблиці 4.1 та рисунках 4.3–4.4 наведено результати виводу з туманом та шумом датчиків для зростання випадковості середи. Алгоритми навчалися у нормальній середі без туману та шуму датчиків, тому ці результати можуть показати швидкодію та надійність агента у стохастичній оточуючій середі. Вивід виконується три кожного алгоритму для отримання середнього та стандартного

відхилення результату. Кожний вивід виконується для 100000 кроків.

Таблиця 4.1 демонструє середнє значення та стандартне відхилення швидкодії кожного алгоритму. На рисунку 4.3 наведена середній показник швидкості, а на рисунку 4.4 показує середню кількість змін смуг руху під час логічного виводу із за туману та шуму датчиків.

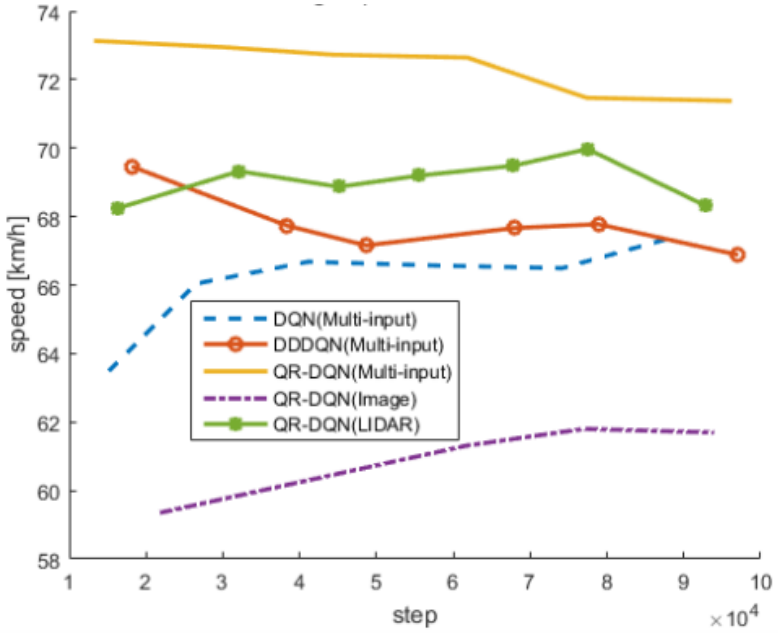

Рисунок 4.3 – Середня швидкість за 3 симуляції

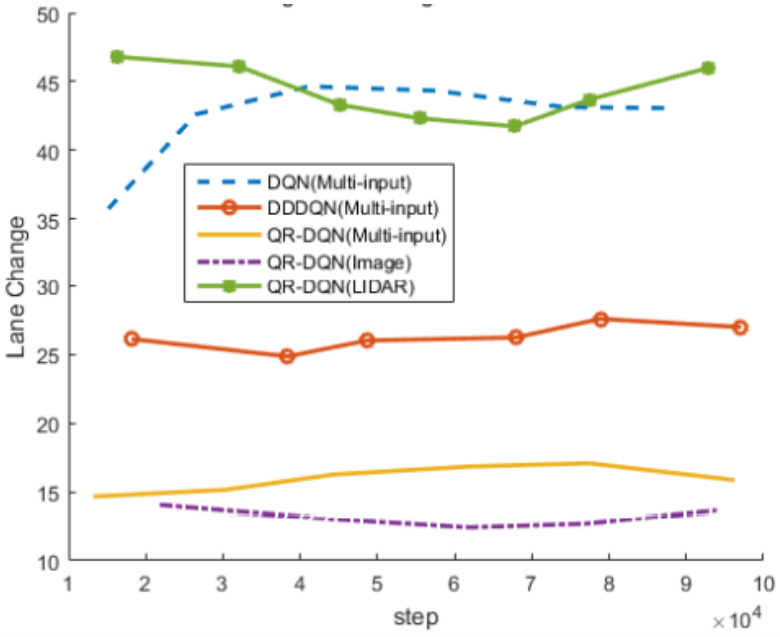

Рисунок 4.4 – Середня кількість змін смуги руху за 3 симуляції

По-перше, алгоритми, мають різні кінцеві дані, які можна порівняти у таблиці 4.1. Середня швидкість QR-DQN з даними LIDAR становить 69.09 км/год. Однак число зміни смуг руху QR-DQN з даними LIDAR становить 44.28, яке є найвищім значенням. Це демонструє, що датчик погіршує дані LIDAR, щоб їх було важче розпізнати. Середня швидкість QR-DQN з вхідними даними у вигляді зображення з камери складає 60.92 км/год, що є найнижчим значенням серед усіх алгоритмів. Однак, середня кількість змін смуг руху дорівнює 13.18, це найменше значення серед усіх алгоритмів. Під час симуляції QR-DQN з вхідними даними у вигляді зображення зазвичай знаходився за переднім автомобілем навіть якщо машин збоку нема, а транспортний засіб попереду їде повільно. Проблема полягає у тому, що QR-DQN з камерою демонструє низьку середню швидкість та кількість змін смуг руху. З іншого боку QR-DQN з сенсорами та камерою демонструє більш високу середню швидкість, ніж QR-DQN з сенсорами та може бути порівняним за середньою кількістю зміни смуг руху з QR-DQN який використовую зображення з камери. Швидкість та зміна смуг руху QR-DQN з декількома вхідними параметрами відповідно 72.38 км/год та 15.98. З цього можна зробити висновок що комбінування структури з декількома вхідними параметрами максимізує швидкодію агента.

По-друге, швидкодія різних алгоритмів DRL з декількома параметрами входу порівнюється у таблиці 4.1. Середня швидкість DQN складає 66.1 км/год, а середня кількість зміни смуг руху дорівнює 42.25. Це демонструє що агент який був навчений на DQN алгоритмі, виконує нестабільний рух з меншою швидкістю, ніж агент DDDQN та QR-DQN. Середня швидкість та середня кількість зміни смуг руху для агента DDDQN відповідно дорівнює 67.78 км/год та 26.31. Цей результат демонструє, що агент навчений за допомогою DDDQN виконує зміну смуг руху меншу кількість разів порівнюючи з агентом навченим на DQN. Тим не менш він демонструє аналогічну середню швидкість у порівнянні з DQN агентом. У випадку QR-DQN показники середньої швидкість та кількості зміни смуг руху є найкращими.

5 АНАЛІЗ МОЖЛИВИХ ЗАСТОСУВАНЬ

Система автомобільної безпеки та допомоги водію стали головною метою для технологій вбудованих систем у реальному часі. Системи майбутнього стрімголов переходять на цю технологію. На ринку ADAS демонструється стабільне зростання. Очікується, що у найближчому майбутньому він обійде все інші автомобільні програмні забезпечення. Спостерігається постійний перехід від повністю механічного до чисто електронному автономного водіння. До 2030 року майже кожне авто на дорозі буде повністю автономним з технологією ADAS, яка буде ефективно справлятися зі своїми задачами. Автономні автомобілі будуть переважати на дорозі. Щоб все це стало реальністю, виробники авто вкладають великі засоби у вигляді грошей та технологій, щоб домінувати в автомобільному секторі. Одна помилка, яка не була зафіксована на етапі тестування може привести до дуже великих втрат задля її усунення на більш пізніх етапах. Тому розробка високопродуктивних інструментів ведеться в області автомобільних вбудованих систем.

Дослідження наведені у цій роботі можуть допомогти у розробці та тестуванні нового програмного забезпечення ADAS, так як середовище було покращене, порівняно з бакалаврською роботою, та збільшена випадковість дій у середі за допомогою щільності автомобільного трафіку, туману та сенсорного шуму. Тому, якщо довершити програму можливо отримати повністю автономне середовище для тестування ADAS без необхідності виходити за рамки комп’ютерної симуляції на ранніх етапах, а також за допомогою новітніх алгоритмів RL, покращити вже існуюче програмне забезпечення.

ВИСНОВКИ

Під час виконання атестаційної роботи був застосований алгоритм QR-DQN, який є одним з розподілених алгоритмів навчання з підкріпленням, з декількома параметрами вводу, що дало змогу покращити швидкодію агента порівняно з бакалаврською роботою. В звичайних системах допомоги водію дії для запобігання зіткнення включені для забезпечення безпеки водія. На основі мереж з декількома вхідними параметрами для посилення сприйняття та збільшення швидкодії було проведено порівняння. Результат отриманий за допомогою розподіленого навчання з підкріпленням продемонстрував найбільшу швидкодію порівняно з іншими алгоритмами з не розподіленим підходом, які детерміновано оцінювали очікувані майбутні винагороди.

Був проведений аналіз існуючих шляхів вирішення проблеми навчання нейронної мережі з підкріпленням для створення автопілоту. Був проведений аналіз методів навчання з підкріпленням, та детально розглянуті алгоритми рішення та описані його недоліки та переваги, що дало можливість оптимальний шлях вирішення їх проблеми.

Це дослідження підтвердило можливість використання глибокої розподіленої архітектурного алгоритму для навчання супервізора(агента) управлінню автономним автомобілем та запропонувало новий напрям дослідження відносно автономного транспорту та покращення навчання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Silver D., Huang A., Maddison C. J., Guez A. [et al] Mastering the game of go with deep neural networks and tree search // Nature. 2016. P. 484.

2. Silver D., Schrittwieser J., Simonyan K., Antonoglou I. [et al] Mastering the game of go without human knowledge // Nature. 2017. P. 354.

3. Mnih V., Kavukcuoglu K., Silver D., Rusu A. A. [et al] Human-level control through deep reinforcement learning // Nature. 2015. P. 529.

4. Finn C., Tan X. Y., Duan Y., Darrell T. [et al] Deep spatial autoencoders for visuomotor learning // IEEE. 2016. P. 512–519.

5. Visual reinforcement learning with imagined goals. URL : https://arxiv.org/abs/1807.04742 (дата звернення: 13.11.2020).

6. Proximal policy optimization algorithms. URL : https://arxiv.org/abs/1707.06347 (дата звернення: 07.12.2020).

7. Schulman J., Levine S., Abbeel P., Jordan M. [et al] Trust region policy optimization // International Conference on Machine Learning. 2015. P. 1889–1897.

8. A distributional perspective on reinforcement learning. URL : https://arxiv.org/abs/1707.06887 (дата звернення: 08.12.2020).

9. Distributional reinforcement learning with quantile regression. URL: https://arxiv.org/abs/1710.10044 (дата звернення: 27.10.2020).

10. Implicit quantile networks for distributional reinforcement learning. URL: https://arxiv.org/abs/1806.06923 (дата звернення: 27.10.2020).

11. Rausch V., Hansen A., Solowjow E., Liu C. [et al] Learning a deep neural net policy for end-to-end control of autonomous vehicles // ACC. 2017. P. 4914–4919.

12. John V., Mita S., Tehrani H., and Ishimaru K. Automated driving by monocular camera using deep mixture of experts // IEEE. 2017. P. 127–134.

13. Van Hasselt H., Guez A., and Silver D. Deep reinforcement learning with double q-learning // AZ. 2016. P. 5.

14. Koenker R. and Hallock K. F. Quantile regression // Journal of economic

perspectives. 2001. P. 143–156.

15. Al-Qizwini M., Barjasteh I., Al-Qassab H., and Radha H. Deep learning algorithm for autonomous driving using googlenet // IEEE. 2017. P. 89–96.