1

Хмельницький національний університет
Факультет програмування та
комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерної інженерії та системного програмування

ДИПЛОМНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 – Комп'ютерна інженерія

на тему «Інтелектуалізована система на основі методів машинного навчання для розроблення комп'ютерних ігор»

ДРКІ. 01583.16.01.16 ПЗ

Виконав: студент 2 курсу, група КІ2м-19-1 _____ Сергєєв Є.В.
Підпис Ініціали, прізвище

Керівник канд. техн. наук, доцент _____ Гнатчук Є.Г.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІСП, д.т.н., професор
Т.О. Говорущенко
25 05 2021 р.

2

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем

Кафедра Комп'ютерної інженерії та системного програмування

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітньо-професійна програма підготовки магістра

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

"07" 09 2021 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Сергєєву Євгенію Віталійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інтелектуалізована система на основі методів машинного навчання для розроблення комп'ютерних ігор

Керівник проекту (роботи) Гнатчук Є.Г., к.т.н., доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від №7 від 15 січня 2021

2. Строк подання студентом проекту (роботи) на кафедру 24.05.2021

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

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

Огляд існуючих технологій для розроблення комп'ютерних ігор

Огляд існуючих методів машинного навчання

Модель ШІ-прискорювача в сучасних процесорах для обрахунку штучного інтелекту в іграх

Агентно-орієнтований метод, що дозволяє підвищити продуктивність агента

Програмні засоби інтелектуалізованої системи на основі методів машинного навчання для розроблення комп'ютерних ігор

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

3

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., доцент кафедри КІСП		
Антиплагіат	Нічепорук А.О., доцент кафедри КІСП		

7. Дата видачі завдання « 09 » 09 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики ДРМ з керівником	09.09.2020	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	12.10.2020	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	05.11.2020	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	28.11.2020	виконано
5	Робота над науковою статтею	11.01.2021	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	26.02.2021	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	28.03.2021	виконано
8	Оформлення пояснювальної записки згідно вимог	13.04.2021	виконано
9	Попередній захист ДРМ	15.04.2021	виконано
10	Захист ДРМ на засіданні ЕК	24.05.2021	

Студент Підпис Сергєєв Є.В. Ініціали, прізвище

Керівник проекту (роботи) Підпис Гнатчук Є.Г. Ініціали, прізвище

РЕФЕРАТ

Тема дипломної роботи: Інтелектуалізована система на основі методів машинного навчання для розроблення комп'ютерних ігор

Автор роботи: магістр Сєргєєв Є.В.

Керівник роботи: к.т.н., доцент Гнатчук Є.Г.

Пояснювальна записка: 91 с., 32 рис., 3 дод., 95 джерел.

ПЕРЕЛІК КЛЮЧОВИХ СЛІВ: процесор, інтелектуалізована система, інтелектуальний агент, машинне навчання, комп’ютерна гра, штучний інтелект.

Метою дипломної роботи є дослідження інтелектуалізованого підходу з використанням апаратних засобів на основі методів машинного навчання для розроблення комп'ютерних ігор.

Об’єкт дослідження – процес прискорення обрахунку штучного інтелекту під час машинного навчання при розробці комп'ютерних ігор.

Предмет дослідження – інтелектуалізований підхід з використанням апаратних засобів на основі методів машинного навчання для розроблення комп'ютерних ігор.

Задачі дослідження:

1) провести огляд існуючих апаратних та програмних технологій, що використовуються для підвищення ефективності штучного інтелекту під час машинного навчання;

2) провести огляд існуючих методів машинного навчання при розробці комп'ютерних ігор;

3) запропонувати технологію вибору апаратного та програмного забезпечення методів машинного навчання при розробці комп'ютерних ігор;

4) на основі запропонованої технології розробити програмні засоби.

Наукова новизна отриманих результатів:

1) Набула подальшого розвитку модель ШІ-прискорювача в сучасних процесорах для обрахунку штучного інтелекту в іграх, що використовує процесор Intel I9 11900 (11 покоління).

2) Удосконаленний агентно-орієнтований метод, що дозволяє підвищити продуктивність агента за рахунок врахування вагового коефіцієнта винагороди. що в свою чергу задовільняє функціональну та дизайнерьску систему гри.

На основі запропонованого підходу розроблена інтелектуалізована система з використанням процесора Intel I9 11900 (11 покоління), що дозволяє при застосуванні методів машинного навчання для розроблення комп'ютерних ігор, підвищити ефективність та швидкість навчання персонажів ігрових проектів.

Практична значимість отриманих результатів полягає у тому, що отримані результати магістерської роботи можуть бути використані для підвищення ефективності використання машинного навчання при розробці комп’ютерних ігор.

Зв’язок роботи з науковими програмами, планами, темами. Дослідження, представлені у кваліфікаційній роботі, проводились в рамках держбюджетної НДР Хмельницького національного університету № 1Б-2019 «Агентно-орієнтована система підвищення безпеки та якості програмного забезпечення комп’ютерних систем» (номер державної реєстрації 0119U100662).

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АНМ – апаратні нейронні мережі

БД – база даних

БЗ – база знань

ГА – генетичні алгоритми

ІС – інтелектуалізовано система

МН – машинне навчання

НН – нейронна мережа

ПЗ – програмне забезпечення

ПЛІС – програмована логічна інтегральна схема

ШІ – штучний інтелект

ШНМ – штучні нейронні мережі

API – прикладний програмний інтерфейс

ASIC – інтегральні схеми спеціального призначення

FPGA – програмована користувачем вентильна матриця

GPU – графічний процесор

ВСТУП

Популярність комп‘ютерних ігор зростає з кожним роком, ніша ринку дозвілля постійно потребує поповнень новими ігровими рішеннями. Ігрова індустрія постійно розробляє нові інструментальні засоби для полегшення та пришвидшення процесу розробки ігор.

Штучний інтелект в комп’ютерних іграх відноситься до адаптивного штучного інтелекту.

Штучний інтелект це певний двигун, що визначає поведінку ігрових персонажів в ігровому світі комп’ютерних ігор. В такому випадку ігрові персонажі діють розумно та креативно, ніби ними керує людина-гравець. Хоча штучний інтелект в тій чи іншій формі давно з’явився і відеоіграх, він вважається бурхливим новим рубежем в тому як ігри розробляються та як в них грають. Ігри з штучним інтелектом дедалі більше переносить контроль над ігровим досвідом у бік гравця, в свою чергу, поведінка якого допомагає створити новий ігровий досвід. Процедурна генерація штучного інтелекту також відома як процедурне «розповідання історій», що в ігровому дизайні відноситься до ігрових даних, що генеруються алгоритмічно, а не спеціально до кожного елементу, що будується спеціально розробником. Основна задача штучного інтелекту в іграх – це покращення досвіду гравця. Це особливо важливо, оскільки розробники забезпечують ігровий досвід на різних пристроях. Компютерні ігри це більше не просто вибір між ігровою консоллю або настільним комп’ютером. Гравці очікують отримання захоплюючого ігрового досвіду на великому наборі мобільних та десктопних пристроїв, від смартфонів до гарнітур доповненої реальності та іншого. Штучний інтелект дозволяє розробникам надавати такий досвід на різних типах пристроїв. Велика різноманітність, конкурентоздатність, специфічність використаних технологій роблять задачу вибору методів при розробці ігор актуальною.

Метою дипломної роботи є дослідження інтелектуалізованого підходу з використанням апаратних засобів на основі методів машинного навчання для розроблення комп'ютерних ігор.

Об’єкт дослідження – процес прискорення обрахунку штучного інтелекту під час машинного навчання при розробці комп'ютерних ігор.

Предмет дослідження – інтелектуалізований підхід з використанням апаратних засобів на основі методів машинного навчання для розроблення комп'ютерних ігор.

Задачі дослідження:

1) провести огляд існуючих апаратних та програмних технологій, що використовуються для підвищення ефективності штучного інтелекту під час машинного навчання;

2) провести огляд існуючих методів машинного навчання при розробці комп'ютерних ігор;

3) запропонувати технологію вибору апаратного та програмного забезпечення методів машинного навчання при розробці комп'ютерних ігор;

4) на основі запропонованої технології розробити програмні засоби.

Наукова новизна отриманих результатів:

1) Набула подальшого розвитку модель ШІ-прискорювача в сучасних процесорах для обрахунку штучного інтелекту в іграх, що використовує процесор Intel I9 11900 (11 покоління).

2) Удосконалений агентно-орієнтований метод, що дозволяє підвищити продуктивність агента за рахунок врахування вагового коефіцієнта винагороди. що в свою чергу задовольняє функціональну та дизайнерську систему гри.

На основі запропонованого підходу розроблена інтелектуалізована система з використанням процесора Intel I9 11900, що дозволяє при застосуванні методів машинного навчання для розроблення комп'ютерних ігор, підвищити ефективність та швидкість навчання персонажів ігрових проектів.

Практична значимість отриманих результатів полягає у тому, що отримані результати магістерської роботи можуть бути використані для підвищення

ефективності використання машинного навчання при розробці комп’ютерних ігор.

Зв’язок роботи з науковими програмами, планами, темами. Дослідження, представлені у кваліфікаційній роботі, проводились в рамках держбюджетної НДР Хмельницького національного університету № 1Б-2019 «Агентно-орієнтована система підвищення безпеки та якості програмного забезпечення комп’ютерних систем» (номер державної реєстрації 0119U100662).

За темою дипломної роботи подано статтю «Using artificial intelligence accelerators to train computer game characters» (подано в журнал Computer Systems and Information Technologies).

1 АНАЛІЗ ВІДОМИХ РІШЕНЬ

1.1 Аналіз апаратного забезпечення, що використовує штучний інтелект для прискорення навчання в комп’ютерних іграх

Інструменти штучного інтелекту (ШІ) та машинного навчання набули значного поширення в останні роки, завдяки досягненням обчислювальних систем в потужності, обсягів обчислень та продуктивності. Штучний інтелект використовується в надзвичайно широкому спектрі програм для отримання кращих результатів порівняно з традиційними методами. До таких додатків відноситься обробка зображень, таких як виявлення та розпізнавання обличчя [1], аналіз фінансових ринків та банківська справа [2], роботизовані комплекси в промисловості [3], медичні додатки та додатки охорони здоров’я [4], ефективні транзакції в управлінні базами даних [5], додатки безпеки [6], безпілотні служби доставки та особистий транспорт [7], автономні безпілотники для навігації [8] та багато іншого. Враховуючи вимоги таких додатків до точності та ефективності, у багатьох з цих додатків в основі лежить штучний інтелект. Ця тенденція свідчить про постійний інтерес та високий потенціал інструментів штучного інтелекту та машинного навчання. Ці інструменти все частіше стають невід’ємною частиною кожної електронної або вбудованої системи. Мікроботи з вбудованим штучним інтелектом широко використовуються на практиці в різних галузях. Роботи мають в своєму складі вдосконалені контролери, які реалізують функції ШІ, такі, наприклад, як сприйняття (отримання інформації) та пізнання (прийняття рішень).

Однією з головних проблем, з якими стикаються розробники таких систем та додатків, є те, що алгоритми штучного інтелекту є обчислювально дорогими. Цей факт змушує шукати шляхи ефективного вирішення цієї проблеми, зокрема використання поліпшення апаратного прискорення, щоб забезпечити необхідну високу обчислювальну потужність. Оптимізована та спеціалізована апаратна реалізація може зменшити системні витрати за рахунок оптимізації необхідних ресурсів та зменшення вимог до енергії при одночасному поліпшенні продуктивності [9].

Як правило, алгоритми ШІ розробляються та тестуються з використанням платформ розробки, таких як TensorFlow, Keras та Caffe. Деякі з цих платформ є популярними стандартними нейронними мережами. Потім розроблені нейронні мережі для конкретної групи застосувань реалізуються в апаратному забезпеченні та називаються апаратними прискорювачами. Впровадження апаратних засобів розглядається на ПЛІС, графічних процесорах та інтегральних схемах спеціального призначення (ASIC), кожна з яких має свої переваги та недоліки. Ці апаратні прискорювачі використовують паралельність для збільшення пропускної здатності та забезпечують набагато вищу продуктивність у порівнянні з традиційними процесорами, які можуть бути значно повільнішими. Алгоритми штучного інтелекту, особливо алгоритми глибокого навчання, потребують значного запасу для використання великих баз даних та потужної обчислювальної обробки для створення автономних адаптивних інтелектуалізованих систем для високоточної та подібної до людини поведінки [10]. Слід зазначити, що галузь глибокого навчання з'явилась лише у 2006 році. Завдяки своїм особливостям, глибоке навчання має суттєві переваги в порівнянні з традиційними методами, що використовують системи, особливо в комп'ютерних іграх. Використання глибокого навчання забезпечувало високу точність, усуваючи фактор людської помилки.

В роботі [11] проведено дослідження апаратної реалізації алгоритмів штучного інтелекту та машинного навчання з 2009 року по 2019 рік. Основною метою цієї роботи було впровадження нейронних мереж як інструменту для виявлення та розпізнавання об'єктів у різних додатках. В цьому дослідженні проаналізовані та приведені результати двісті однієї наукової роботи, 169 робіт стосувались апаратної реалізації алгоритмів штучного інтелекту та машинного навчання. Апаратне прискорення вважається ідеальним рішенням для енергоємних та ресурсозатратних алгоритмів штучного інтелекту. Метою є досягнення більш швидкої та ефективної обробки алгоритмів ШІ [12]. За останні роки опубліковано багато досліджень, в яких обговорюється велика кількість реалізацій апаратного та програмного забезпечення оптимізації та методів

реалізації в цій галузі [13-20]. Частина досліджень розглядала впровадження штучних нейронних мереж на апаратному забезпеченні загалом, інші дослідження були зосереджені на прискорювачах FPGA для нейронних мереж глибокого навчання. Наприклад, у роботі [15] автори зосередилися на реалізації FPGA загорткових нейронних мереж. В роботі [17] в ході дослідження обговорювались деталі реалізації графічного процесору GPU.

В роботі [21] досліджено усі основні апаратні реалізації штучних нейронних мереж, реалізованих на апаратному забезпеченні, що називаються апаратними нейронними мережами (АНМ). Апаратні нейронні мережі класифікували за деякими ознаками, такими як вбудована/відключена мікросхема, аналогові/цифрові блоки, порогові значення, таблиця пошуку, обчислення та швидкість передачі даних. Спеціальний розділ для мікросхем АНМ охоплює реалізації цифрових та гібридних нейрочіпів на основі FPGA та ASIC.

В роботі [22] проведене більш детальне дослідження, основні аспекти якого були присвячені прискорювачам нейронних мереж на основі FPGA, де вони дослідили конструкції прискорювачів нейронних мереж та підсумували всі методи, що використовуються для автоматизації проектування прискорювачів.

В дослідженні [24] обговорювали основи навчання глибоких нейронних мереж із зазначеним розділом для методів реалізації стиснення. Опитування аналізує прискорювачі на основі FPGA та прискорювачі на базі ASIC, з більшим акцентом на FPGA. Інші дослідження, такі як [25], також розглядали методики проектування прискорювачів на основі FPGA для нейронних мереж стиснення.

Як висновок, можна сказати, що у всі проведені дослідження вивчали різні альтернативи графічного процесора комплексно для покращення продуктивності алгоритмів штучного інтелекту. Використання графічних процесорів є гарним варіантом, що стосується прискорення штучного інтелекту та варіантом, який може скласти конкуренцію рішенням на основі FGPA та ASIC.

Підсумковий огляд розповсюдження платформи апаратного забезпечення в наукових роботах представлений в таблиці 1.1.

Таблиця 1.1 – Огляд розповсюдження платформи апаратного забезпечення в наукових роботах

Рік публікації	Дослідження	Опис та сфера застосування
2010	Штучні нейронні мережі в апаратному забезпеченні: огляд двох десятиліть розвитку	У цій оглядовій роботі розглядаються всі основні підходи та моделі апаратних нейронних мереж протягом 1990–2010 років. Вони надають приклади різноманітних реалізацій АНМ у широкому діапазоні моделей ШНМ, таких як згорткові нейронні мережі, нейронна мережа, що підключається, тощо. Ці АНМ реалізації цифрові, аналогові, гібридні, нейроморфні, ПЛІС або оптичні. Кожен з них обговорюється окремо в різних розділах
2013	Огляд використання програмного та апаратного забезпечення у штучних нейронних мережах	Дослідження полягало в тому, чи використовують розробники готові відкриті джерела програмного забезпечення та апаратних прискорювачів або розробляти власні власні версії. Також представлена таблиця програмних плат, що використовуються для штучних нейронних мереж
2017	Огляд прискорювача нейронних мереж на базі FPGA	Це дослідження дає огляд попередньої роботи над нейронною мережеюприскорювачів висновків на основі ПЛІС та узагальнюють основні використовувані методи
2017	Ефективна обробка глибинних нейронних мереж: підручник та огляд	Ця стаття зосереджується лише на типі ANN глибоких нейронних мереж. У ньому згадується історія, компоненти та програми із апаратним забезпеченням, яке підтримує операції типу DNN
2018	Огляд прискорювачів на основі FPGA для згорткових нейронних мереж	Обговорюються кілька стратегій оптимізації, що використовуються в апаратних архітектурах для CNN, такі як переупорядкування циклу, розгортання та конвеєризація, формат з фіксованою точкою тощо. Автори також класифікують усі реалізації відповідно до підходів оптимізації для архітектури CNN та FPGA або оптимізації пам'яті
2018	Еволюція пристрою графічного процесора, який широко використовується в ШІ та масивній паралельній обробці	Стаття представляє коротку дискусію про історію процесорів та графічних процесорів, детально описуючи закон Мура. Він також містить кілька важливих програм для ШІ

Кінець таблиці 1.1 – Огляд розповсюдження платформи апаратного забезпечення в наукових роботах

Рік публікації	Дослідження	Опис та сфера застосування
2018	Опитування прискорювачів глибокого навчання на основі FPGA: виклики та можливості	У статті представлено опис конструкцій прискорювачів глибокого навчання на базі FPGA. Вони класифікують роботу відповідно до конструкції або для певного алгоритму, або для конкретного застосування, або для універсального прискорювача фреймворк із апаратними шаблонами
2019	Прискорювачі мереж глибокого навчання на основі FPGA для навчання та класифікації: огляд	У статті дається вичерпна довідка та історія прискорювачів на основі FPGA. Огляд включає програми глибокого навчання та обговорює приклади реалізації графічного процесора, ASIC (з конкретними прикладами) та прикладів ПЛІС у детальному аналізі та тестуванні

На рисунку 1.1 представлені результати розповсюдження платформ апаратного забезпечення в інтелектуалізованих системах для створення комп’ютерних ігор.

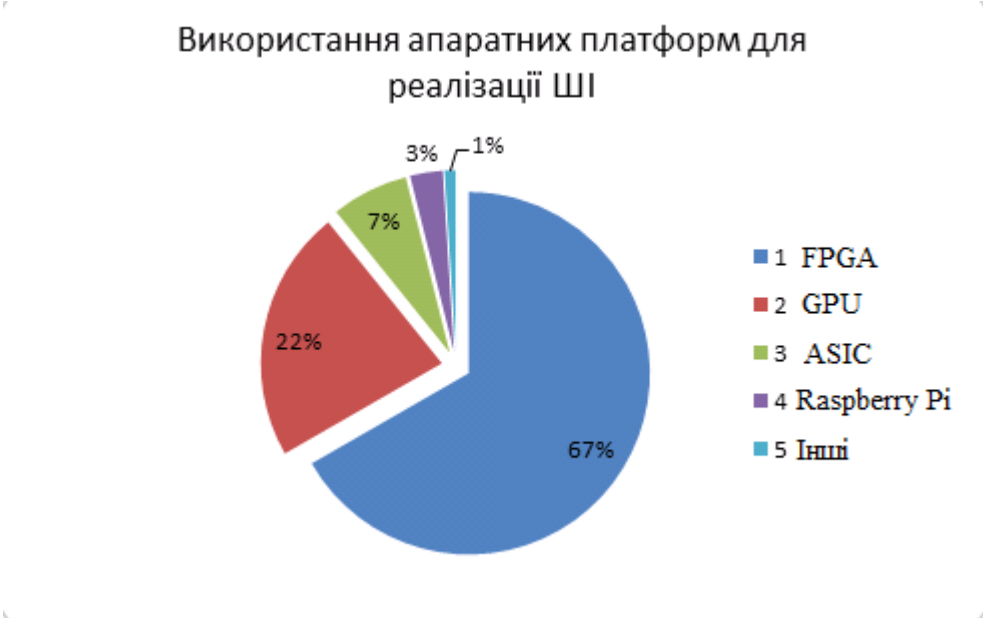


Рисунок 1.1 – Розповсюдження платформ апаратного забезпечення в інтелектуалізованих системах для створення комп’ютерних ігор

Як показано на рисунку 1.1, більшість досліджень присвячені реалізаціям на основі FPGA. Завдяки швидкому середовищу створення прототипів значна кількість досліджень використовувала ПЛІС для реалізації складних алгоритмів ШІ. Здатність FPGA забезпечити просте середовище прототипування з потужними обчислювальними ресурсами зробило дуже корисним для дослідників розробку та тестування нових прискорювачів для алгоритмів штучного інтелекту. FPGA з його високою гнучкістю вважається життєздатним ріщенням для впровадження таких алгоритмів, як глибокі нейронні мережі з нерегулярним паралелізмом та користувацькими типами даних. Більш того, нещодавні тенденції проектування FPGA призвели до того, що вони стали більш доступними та отримали значну увагу для глибоких досліджень. Спеціальна інтегральна схема ASIC також є однією з потужних платформ, що використовуються для реалізації алгоритмів ШІ. Блоки графічної обробки також використовуються для прискорення алгоритмів ШІ, це пришвидшення становить від декількох разів до декількох сотен разів. Графічні процесори призначені для виконання інтенсивних скалярних та паралельних обчислень. На відміну від багатоядерних процесорів, графічні процесори не покладаються на приховані затримки, використовуючи великий обсяг кеш-пам’яті при доступы до пам’яті DRAM. Це призводить до агресивного явного паралелізму, який набагато більше підкреслює оптимізацію пропускної здатності, ніж ядрі центрального процесора. Ці характеристики роблять графічні процесори дедалі кориснішими для прискорення штучного інтелекту.

1.2 Аналіз інтелектуалізованих систем при розробленні комп’ютерних ігор

Історично склалося, що контролери базуються на центральних процесорах загального призначення і лише нещодавно досліджено, що є варіанти використання систем на кристалі або систем на чипі, так званих SoC. Це електронні схеми, які в собі вміщують всі функцій ні складові одного пристрою, використовуючи при цьому одну мікросхему. Дотепер не повідомлялося про персональні комп’ютери з когнітивними можливостями, навіть незважаючи на те, що когніція є ключовою функцією штучного інтелекту в мікроботах для прийняття рішень. Особливо це стосується автономних дронів. Для швидкого реагування для планування шляху та уникнення перешкод вимагається понад 10000 пошуків протягом 50 мс. Для прийняття рішень в таких складних ситуаціях авторами роботи [35] пропонується впровадження програмного забезпечення, яке працює на мікросхемі Cortex-M3 та займає приблизно 5 секунд. Мікросхема Cortex-M відноситься до групи 32-розрядних процесорних ядер RISC ARM, що ліцензовані Arm Holdings. Мікроботи потребують в 10 разів нижчої потужності та в 100 разів швидшого прийняття рішень, ніж звичайні роботи, завдяки швидкому переміщенню в навколишньому середовищі, малому форм-фактору та обмеженій ємності акумулятора. Отже, надмірно потужний високопродуктивний процесор штучного інтелекту необхідний мікроботам для швидких та розумних маневрів в динамічних середовищах, заповнених перешкодами.

Різні програми використовують ансамблеві моделі навчання, використовуючи колекцію дерев рішень, для швидкої та точної класифікації вхідних даних на основі його вектора особливостей. Автори роботи [36] розглядають реалізацію метода випадкових лісів (Random Forests) як першого алгоритму машинного навчання, що виконується на процесорі автоматичних даних. Це реконфігурований прискорювач співпроцесора, який підтримує паралельне виконання численних автоматів проти одного вхідного потоку даних. Завдяки цій моделі виконання, запропонований авторами підхід принципово відрізняється від інших, перекладаючи моделі довільного лісу з існуючих

алгоритмів обходу дерева, пов’язаних пам’яттю, на конвеєрні конструкції, які використовують безліч автоматів для незалежної та паралельної перевірки всіх необхідних порогів. Також описані методи обробки значень функції з плаваючою комою, які не підтримуються у власному обладнанні, конвеєризації етапів виконання та стиснення автоматів для найшвидшого часу виконання. Кінцевий результат – це рішення, яке при оцінці за допомогою двох додатків, а саме розпізнавання рукописних цифр та аналіз настроїв призводить до 63 та 93 разів прискорення відповідно до одноядерних найсучасніших рішень на основі процесора. Автори вважають, що розглянуті алгоритмічні прийоми будуть корисними не тільки для прискорення інших програм, що використовують метод випадкових лісів, але й для реалізації інших методів машинного навчання в цій новій архітектурі.

Алгоритм пошуку шляхів вирішує проблему пошуку найкоротшого шляху від джерела до пункту призначення та уникнення перешкод. Однією з найбільших проблем у розробці реалістичного штучного інтелекту в компютерниї іграх є рух агента. Стратегії пошуку шляхів зазвичай використовуються як ядро будь-якої системи руху штучного інтелекту в комп’ютерних іграх. У роботі [37] алгоритм пошуку використовується для пошуку найкоротшого шляху між джерелом та пунктом призначення на зображенні, яке представляє карту або лабіринт. Пошук шляху через лабіринт - основна проблема інформатики, яка може набувати різних форм. Алгоритм A* широко використовується для пошуку шляхів та обходу графіків. Для перевірки продуктивності системи використовуються різні зображення карти та лабіринту (по 100 зображень на кожну карту та лабіринт). Загальна продуктивність системи є прийнятною і дозволяє знайти найкоротший шлях між двома точками на зображенняї. Понад 85% зображень можуть знайти найкоротший шлях між двома точками.

В комп’ютерних іграх визначення шляхів протягом останніх десятиліть було одним з основних напрямків досліджень. В роботі [38] розглядаються алгоритми пошуку шляхів, що використовуються в жанрі платформ енних відеоігор, та зосереджуються на визначенні найбільш підходящих алгоритмів, які

будуть використовуватись. Для того, щоб визначити алгоритми, що будуть використовуватись в плтформенних іграх, по-перше, потрібно визначити існуючі алгоритми пошуку шляхів, які використовуються на сьогоднішній день. Потім кількість алгоритмів звужується до тієї кількості,що найбільш використовуються через їх переваги перед іншими алгоритмами. Отримані дані дослідження вказують на те, що алгоритми A* та алгоритм Дейкстри є найбільш підходящими алгоритмами для використання. Однак минулі дослідження не включали дослідження вертикальних рухів у 2D – платформах. Тому, виникають питання щодо того наскільки великою буде різниця з точки зору ефективності, якщо ці алгоритми будуть протестовані на цьому конкретному середовищі.

Програми для комп'ютерних ігор зазвичай пропонують гарний досвід роботи під час роботи з десктопними застосунками. Потужні високопродуктивні процесори, що працюють без енергетичних обмежень успішно справляються з дослідженнями великих ігрових дерев, забезпечуючи потужну гру для задоволення вимогливих користувачів. Однак, сьогодні все більше і більше гравців запускають ці ігри на смартфонах иа планшетах де менша обчислювальна потужність і обмежений бюджет енергії надають користувачу набагато слабшу гру. Останні системи на чіпі містять програмовану логіку, що тісно поєднана з процесорами загального призначення, що дозволяє включати спеціальні прискорювачі для будь-яких додатків для підвищення продуктивності та ефективності. В роботі [39] автори аналізують переваги розподілу ШІ десктопних ігор на програмне та апаратне забезпечення. В якості дослідження були обрані три популярні та складні гри Reversi, Blokus та Connect 6. Проаналізовані конструкції включають апаратні прискорювачі для обробки платою, які суттєво покращують продуктивність та енергоефективність, що, в свою чергу, призводить до можливості використовувати набагато більш потужні та акумуляторні програми. Результати дослідження демонструють, що використання апаратного та програмного кодового дизайну для розробки десктопних ігор дозволяє підтримувати або навіть покращувати взаємодію з користувачами на різних платформах, зберігаючи при цьому низьку потужність та енергію.

Протягом останніх років машинне навчання, а особливо його різновид глибоке навчання бурхливо розвивається. Методи, розроблені в цих двох областях можуть аналізувати та вивчати величезну кількість реальних прикладів у різних форматах. Хоча кількість алгоритмів машинного навчання велика і зростає, їх реалізація через фреймворки та бібліотеки теж збільшується. Розробка програмного забезпечення в цій галузі швидко розвивається завдяки великій кількості програмного забезпечення з відкритим кодом. Дослідження [40] представляє великий огляд з порівнянням, а також тенденціями розвитку та використання передового програмного забезпечення для штучного інтелекту. Також приведений огляд масивної підтримки паралелізму, яка здатна ефективино масштабувати обчислення в епоху великих даних.

За результатами проведеного огляду існуючих рішень основні результати приведені в таблиці 1.2.

Таблиця 1.2 – Опис проблем, що виникають при розробленні комп’ютерних ігор

№	Проблема	Опис
1	Складні просторів прийняття рішень	Більшість найсучасніших комп’ютерних ігор пов’язані зі складною стратегічною (стратегічні ігри в реальному часі) або правдоподібною поведінкою (інтерактивні драми). Два типи поведінки поділяють характерні риси отримання величезних просторів для прийняття рішень.
2	Авторська підтримка	Створена вручну поведінка в кінцевому рахунку є програмним кодом на складній мові програмування, на яку поширюються людські помилки.
3	Непередбачувані ситуації	Неможливо підготуватися до всіх ймовірних ситуацій та стратегій гравців, які можуть виникнути під час гри.

Кінець таблиці 1.2 – Опис проблем, що виникають при розробленні комп’ютерних ігор

4	Інженерія знань	Навіть якщо припустити, що стратегії чи поведінка створені вручну, створення таких типів поведінки в грі вимагає величезних людських інженерних зусиль.
5	Можливість відтворення та мінливість	Гравцеві може нудно бачити одні і ті ж самі стратегії та поведінку знову і знову.
6	Риторична мета	Цілком можливо, що розроблена людиною поведінка або стратегія не досягають цілей гри повністю.

1.3 Огляд існуючих методів машинного навчання

Загальний термін машинне навчання об’єднує в собі багато математичних, статистичних та обчислювальних методів для розробки алгоритмів, що мають здатність вирішити задачу не прямим способом, а шляхом пошуку закономірностей в різноманітті вхідних даних. Рішення задачі обчислюється не шляхом використання певних формул, а по встановленій залежності результатів від конкретного набору вхідних ознак та їх значень.

Машинне навчання застосовується в широкому колі галузей, зокрема: для діагностування, прогнозування, розпізнавання та прийняття рішень в різноманітних прикладних галузях від медицини до банківської діяльності.

Існує багато методів машинного навчання в залежності від типу задачі, яку вони вирішують. Але в загальному існує три види машинного навчання:

1. З вчителем, коли необхідно знайти функційну залежність результату від вхідних даних і побудувати алгоритм, що на вході приймає опис об’єкту, а на виході видає відповідь. Функціонал якості, як правило, визначається через середню похибку відповідей алгоритму по всім об’єктам вибірки.

2. Без вчителя, коли відповіді не задаються і потрібно шукати залежності між об’єктами.

3. Навчання з підкріпленням використовується для вирішення більш складних задач, ніж навчання з вчителем та без вчителя. Машина (її часто в такому випадку називають агентом) не має попередньої інформації про середовище, але має можливість виконувати певні дії. Середовище реагує на ці дії і тим самим надає агенту дані, що дозволяють йому реагувати на них та навчатися. Фактично агент та середивище утворюють систему з оберниними зв’язками.

Класифікація методів машинного навчання в залежності від типу задач, які вони вирішують наведена в таблиці 1.3.

Таблиця 1.3 – Класифікація методів машинного навчання в залежності від типу задач, які вони вирішують.

Тип навчання	Вид задачі
Навчання з вчителем	Класифікація Регресія Ранжування Прогнозування
Навчання без вчителя	Кластерізація Пошук асоціативних правил Фільтрація викидів Побудова довірчого інтервалу Скорочення розмірності Заповнення пропущених значень
Навчання з підкріпленням	Співставлення Управління Планування Навчання Навігація

Розглянемо детальніше методи машинного навчання в залежності від типів задач, які вони вирішують.

Навчання з вчителем є найбільш розповсюдженим, популярним та більш широко вивченим методом машинного навчання. Метод навчання з вчителем має працювати з прикладами, що містять не тільки вектор незалежних змінних або атрибутів чи ознак, але й значення, яке має видавати модкль після навчання. Таке значення часто називається цільовим. Різниця між цільовими та фактичними значеннями моделі називається помилкою навчання, або іншими словами похибкою навчання чи нев'язкою, залишком. Вона мінімізується в процесі навчання та виступає в якості вчителя. Значення вихідної помилки потім використовується для обчислення корекцій параметрів моделі на кожній ітерації навчання. В залежності від того, що ми хочемо дізнатись, навчання з вчителем може використовуватись для вирішення двох основних типів задач: задача класифікації та задача регресії. В задачах класифікації в якості цільової змінної використовується мітка класу, а в задачах регресії цільовою змінною є числова змінна цілого або дійсного типу.

Задача класифікації використовується тоді, коли потрібно спрогнозувати дискретні значення, наприклад виконати класифікацію за певними категоріями. Кількість кінцевих відповідей є скінченною.

Задача регресії використовується тоді, коли потрібно спрогнозувати неперервні значення. Цей тип задач не має визначеного обмеження значень, тому що значення може бути будь-яким числом без обмежень.

Завдання та відповідні алгоритми навчання для задач класифікації та регресії наведені на рисунках 1.2 та 1.3 відповідно.

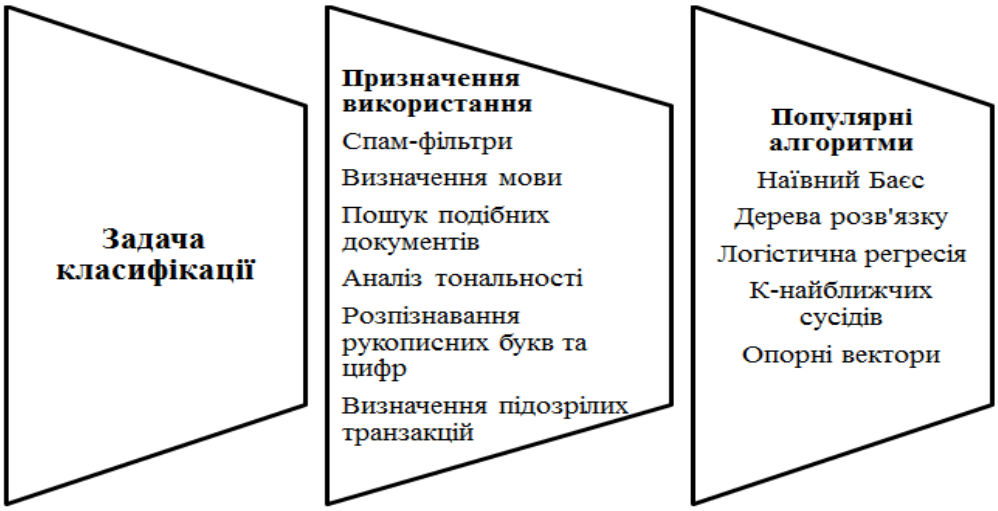


Рисунок 1.2 – Задача класифікації та відповідні алгоритми

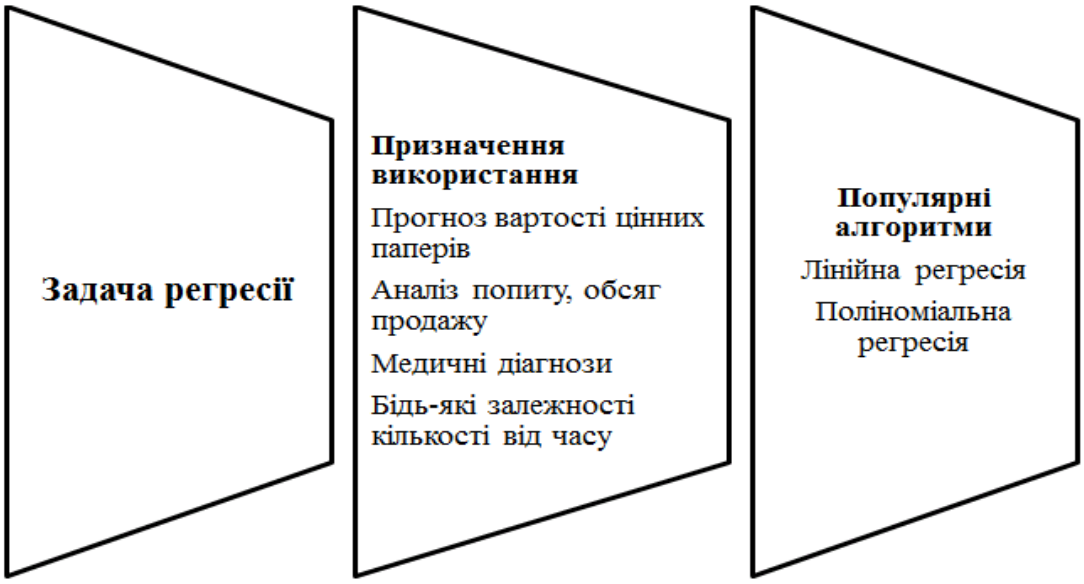


Рисунок 1.3 – Задача регресії та відповідні алгоритми

Ще два типи задач, що відносяться до методів навчання з вчителем це задача ранжування та задача прогнозування. Ранжування – це спосіб сортування елементів на основі їх релевантності. Під релевантністю розуміється ступінь відношення об’єкта до визначеного запиту.

Навчання без вчителя це метод машинного навчання, що не використовує цільову функцію для корекції параметрів моделі, що навчається. В таких алгоритмах вихідна помилка моделі на множині, що навчається не обчислюється. Замість цього використовується інформація про поточний стан параметрів моделі та прикладів множини, що навчається. Наприклад, це може бути Евклідовий простір між векторами ознак прикладу та вектором ваг нейронів, що і буде керувати корекцією параметрів моделі в ході навчання. Навчання без вчителя використовується для ряду задач: задача кластерізації, пошук правил асоціації, скорочення розмірності даних, візуалізація даних. В певній степені кожна з останніх трьох задач є похідною від першої або її частковим випадком.

Під задачею пошуку правил асоціацій мається на увазі виявлення в ознакових описах об’єктів (вихідних даних) таких наборів і значень ознак, які особливо часто зустрічаються у вихідних даних. Якщо проводити анологію з першою задачею, то кожне правило в даному випадку може бути представлене як кластер. Задача скорочення розмірності даних полягає в наступному. Існує великий об’єм ознакових описів об’єктів. Причому той об’єм обумовлений великою кількістю вимірів ознакового простору. Необхідно представити тіж дані в просторі меншої розмірності при цьому мінімізувати втрати інформації. Групування по кластерам як раз і є одним з варіантів вирішення проблеми.

Задача візуалізації даних є по суті частковим випадком попередньої. Її мета представити вихідні дані в просторі, що відображається. Навчання без вчителя тією чи іншою мірою зводиться до кластерізації. Тому для оцінки якості навчання даним способом, як правило, використовують метрики якості кластерізації. При їх виборі враховуються, що ці метрики не повинні залежити від вихідних даних, а тільки від результатів розбиття. Всі оцінки якості можна розділити на зовнішні та внутрішні. Зовнішні використовують зовнішню інформацію про дійсне розбиття об’єктів на кластери, внутрішні спираються тільки на набір вихідних даних, тобто дані метрики можуть працювати з нерозміченою вибіркою коли завчасно відомо істине розбиття об’єктів на групи. І саме з їх допомогою визначається оптимальне число кластерів.

Задача зменшення розмірності в машинному навчанні має на увазі зменшення числа ознак наборів даних. Наявність в ньому ознак надлишкових, неінформативних або слабо інформативних може понизити ефективність моделі. Після такого перетворення модель спрощується і відповідно зменшується розмір набору даних в памяті та прискорюється робота алгоритмів машинного навчання.

Завдання та відповідні алгоритми навчання для задач кластерізації, задачі зменшення розмірності та задачі пошуку правил наведені на рисунках 1.4, 1.5 та 1.6 відповідно.

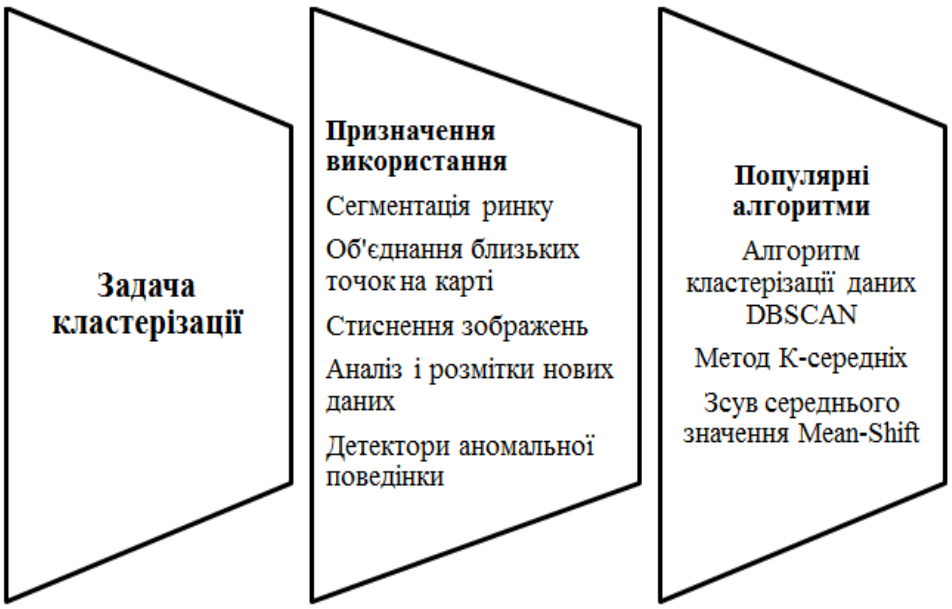


Рисунок 1.4 – Задача кластерізації та відповідні алгоритми

Рисунок 1.5 – Задача зменшення розмірності та відповідні алгоритми

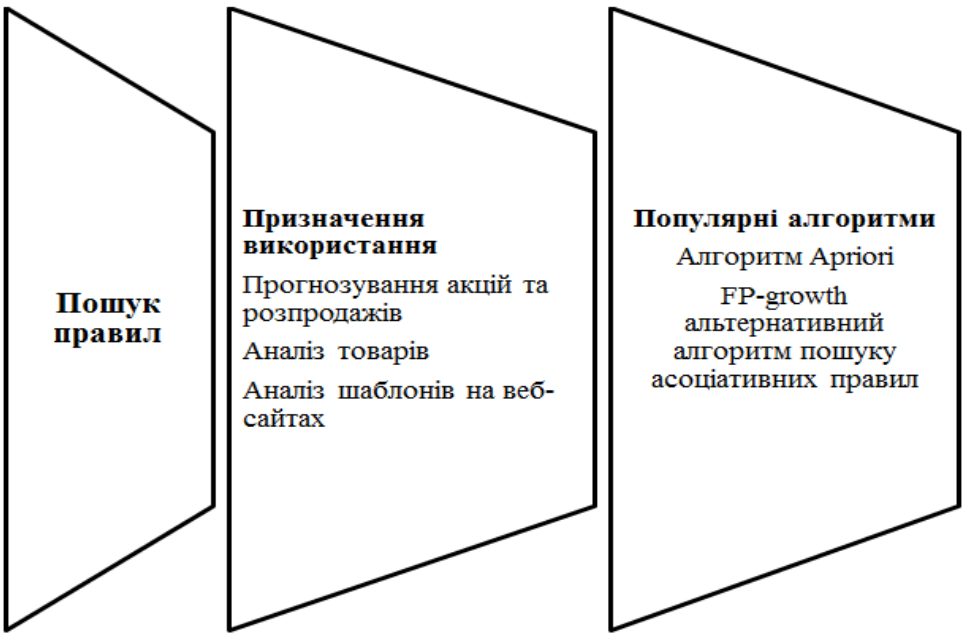


Рисунок 1.6 – Задача пошуку правил та відповідні алгоритми

Навчання з підкріпленням (RL) - це область досліджень, яка надзвичайно розквіла за останні роки і показала надзвичайний потенціал для опонентів на основі штучного інтелекту в комп'ютерних іграх. Цей успіх насамперед

обумовлений величезними можливостями згорткових нейронних мереж, які можуть витягувати корисні функції із галасливих та складних даних. Ігри є чудовими інструментами для тестування та розширення меж нових алгоритмів RL, оскільки вони дають цінне розуміння того, наскільки алгоритм може працювати в ізольованих середовищах без реальних наслідків. Стратегічні ігри в режимі реального часу (RTS) - це жанр, який має надзвичайно складну задачу та кидає виклик гравцю в короткостроковому та довгостроковому плануванні [41]. Існує багато досліджень, які зосереджуються на застосованому RL в іграх RTS, і тому нові досягнення очікуються в не дуже віддаленому майбутньому. Однак на сьогодні існує небагато середовищ для тестування ШІ RTS. Середовища в літературі часто або надто спрощені, такі як microRTS, або складні, і не мають можливості для прискореного навчання на споживчому обладнанні, як StarCraft II. У цій роботі представлено ігрове середовище Deep RTS для тестування передових алгоритмів штучного інтелекту для ігор RTS. Deep RTS - це високопродуктивна гра RTS, створена спеціально для досліджень штучного інтелекту. Він підтримує прискорене навчання, це означає, що він може вчитися в 50 000 разів швидше порівняно з існуючими іграми RTS. Deep RTS має гнучку конфігурацію, що дозволяє проводити дослідження в декількох різних сценаріях RTS, включаючи частково спостережувані простори станів та складність карти. Ми показуємо, що Deep RTS виконує наші обіцянки, порівнюючи свою продуктивність з microRTS, ELF та StarCraft II на високотехнологічному споживчому обладнанні. Використовуючи Deep RTS, ми показуємо, що агент Deep Q-Network перевершує агенти випадкової гри понад 70% часу. Deep RTS є загальнодоступним на https://github.com/cair/DeepRTS.

Завдання та відповідні алгоритми навчання для навчання з підкріпленням наведені на рисунку 1.7.

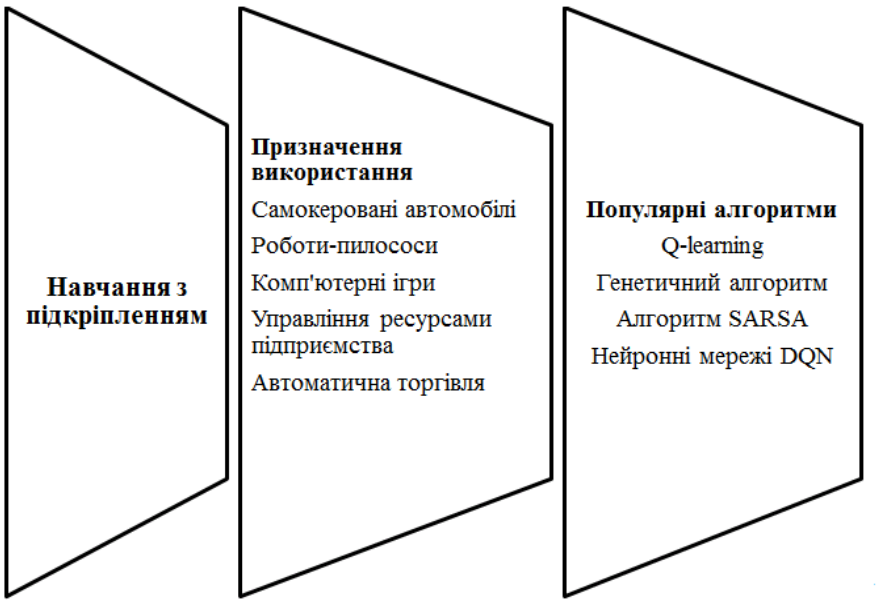


Рисунок 1.7 – Задача навчання з підкріпленням та відповідні алгоритми

Програми, що можуть краще використовувати отримані дані для навчання та виконання корисних дій мають переваги у використанні. Глибокі нейронні мережі, поряд із досягненнями в класичному машинному навчанні та масштабованими обчислювальними пристроями графічного процесора загального призначення GPU стали критично важливими компонентами штучного інтелекту, що дозволило здійснити багато досліджень. Мова Python продовжує залишатись найбільш популярною мовою для наукових обчислень, науки про дані та машинного навчання, підвищуючи продуктивність, так і дозволяючи використовувати бібліотеки низького рівня та чисті API високого рівня. Дослідження [39] пропонує використати підхід до машинного навчання за допомогою Python. В роботі зібрані широко використовувані бібліотеки та концепції для цілісного порівняння їх використання в галузі машинного навчання на Python.

1.4 Постановка задачі

Огляд літератури показав, що на сьогоднішній день зростає популярність комп‘ютерних ігор. Все більше дослідників, зокрема зарубіжних, досліджують галузь та пропонують нові інструментальні засоби для полегшення та пришвидшення процесу розробки ігор.

Враховуючи складність обчислювальних процесів та велику вартість цих процесів, ігрова комп’ютерна галузь потребує вдосконалення апаратного та програмного забезпечення для підвищення ефективності та швидкості обробки алгоритмів штучного інтелекту.

Отже, використання прискорювачів штучного інтелекту для навчання персонажів комп’ютерних ігор за допомогою інтелектуалізованої системи на основі методів машинного навчання є актуальною задачею.

В процесі виконання роботи необхідно розв’язати наступні задачі:

1) провести огляд існуючих апаратних та програмних технологій, що використовуються для підвищення ефективності штучного інтелекту під час машинного навчання;

2) провести огляд існуючих методів машинного навчання при розробці комп'ютерних ігор;

3) запропонувати технологію вибору апаратного та програмного забезпечення методів машинного навчання при розробці комп'ютерних ігор;

4) на основі запропонованої технології розробити програмні засоби.

2 РОЗРОБЛЕННЯ ФОРМАЛІЗОВАНОЇ МОДЕЛІ

2.1 Особливості машинного навчання з підкріпленням в комп’ютерних іграх

На даний момент, штучний інтелект в іграх представляє з себе кінцевий автомат, який на кожну дію гравця змінює свій стан відповідно до записаного заздалегідь правила від розробника. І як би детально над нею не попрацювали, завжди знайдеться умовність або випадок, який гравцю нагадає, що перед ним лише набір "якщо – то".

Навчання з підкріпленням дуже схоже на навчання з учителем, але є певна різниця. Різниця полягає в тому, що в ролі «учителя» виступає або віртуальне або справжнє середовище. Враховуючи, що в роботі розглядаються методи машинного навчання для комп’ютерних ігор, то в подальшому ми будемо говорити про віртуальне середовище. Якщо говорити про комп’ютерні ігри, то навчання з підкріпленням виглядає наступним чином. Наприклад, робота або агента поміщують в певний лабіринт з якого він сам має знайти вихід. Під час цього пошуку робот або агент отримує від зовнішнього, по відношенню до лабіринту, середовища інформацію про те, де вихід відсутній. Таким чином, відбувається вивчення навколишнього світу та знаходження шляху до виходу. В даному випадку нагородою за успішне виконане завдання може бути перехід на новий рівень або можливість отримати нове завдання, а також бали набрані в прцесі виконання завдання. Як правило, чим ефективніше виконане завдання, тим більша кількість балів може бути нарахована. Під ефективністю виконання завдання може розумітися, наприклад, час який робот або агент витратив на пошук виходу з лабіринту. Навчання з підкріпленням використовується в тих випадках, коли необхідно обрати найкращий варіант серед багатьох запропонованих варіантів або домогтися досягнення мети з використанням безлічі ходів. Саме в цьому випадку проявляється штучний інтелект в дії, тобто робот або агент намагається вирішити поставлену задачу, обираючи з безлічі запропонованих варіантів, враховує свої помилки, вчиться на цих помилках і таким чином покращує свої показники. Метод навчання з підкріпленням в першу

чергу застосовується в тих випадках, де потрібно вчити робота або агента діяти в реальному середовищі.

При використанні навчання з підкріпленням можливі дві мети навчання:

1. Першою можна розглянути мету мінімізації помилок. Робот або агент вчиться проводити аналіз інформації перед тим як зробити кожен наступний хід.

2. Другою можна розглянути мету отримання від виконання завдання максимальної вигоди. Сама ця вигода має бути визначена та запрограмована заздалегідь. Вигодою може бути, наприклад, оптимальне витрачання ресурсів, максимальна швидкість проходження по маршруту, мінімальний час витрачення на подолання маршруту чи виконання завдання, обслуговування максимальної кількості відвідувачів, тощо.

Навчання з підкріпленням застосовується там, де необхідно зробити порівняння відкладеної вигоди (іншими словами мети) з прийняттям рішення в певній ситуації. Цей вид навчання покликаний вирішити складне завдання. А семе, як співставити негайні дії з відкладеною віддачею, яку ці дії продукують. В багатьох ситуаціях алгоритми з підкріпленим навчанням мають зачекати, щоб побачити результати прийняття своїх рішень. Недоліком таких випадків є те, що буває складно зрозуміти яка саме дія приводить до того чи іншого результату.

Практичне застосування навчання з підкріпленням найчастіше застосовується для наступних галузей:

- планування;
- постановка цілей;
- боти для комп’ютерних ігор;
- системи зорового сприйняття;
- трейдингові боти;
- чат боти.

У [70] автори розробили евристичний алгоритм пошуку та архітектуру набору команд потоку даних (ISA). Запропонована конструкція використовувала DNNWEAVER для створення високопродуктивних прискорювачів, що працюють в рамках обмеженого бюджету енергії та вбудованої пам'яті FPGA.

DNNWEAVER - це структура, яка автоматично генерує синтезований прискорювач для даної пари (DNN, FPGA) за допомогою оптимізованих вручну шаблонів [90]. У роботі [89] була побудована структура DeepBurning для спрощення операції відображення різноманітних нейронних мереж у ПЛІС або ASIC. У роботі [21] автори запропонували бібліотечний компілятор RTL для автоматичного створення спеціальних прискорювачів на основі FPGA для даного алгоритму CNN. Порівнюючи [21, 21, 89], отримано більше 6,4 прискорення порівняно з [89]. Інші бібліотечні компілятори RTL були запропоновані в [106, 110, 111] для прискорення CNN на платформах FPGA.

У роботі [15] автори досліджували можливість прямого апаратного картографування CNN на ПЛІС. Запропонована модель стверджує, що відкриває нові можливості для подальшої оптимізації і може бути розширена на технології ASIC, а також на двійкові нейронні мережі. Методологія проектування на основі затримок для відображення ConvNets у ПЛІС була запропонована в [17]. Автори [18] розробили засіб автоматизації на основі пітона для створення прискорювачів CNN на основі FPGA. Інструмент постійно забезпечував високу пропускну здатність для різних моделей CNN. В інших роботах, таких як [19], пропонується f-CNNx, який є автоматизованим потоковим потоком для оптимізованого відображення декількох CNN на FPGA.

2.2 Формалізований опис моделі NPC

Створення правдоподібних персонажів є однією з найскладніших проблем в інтерактивній індустрії розваг. Хоча для дизайнерів та програмістів доступні різні інструменти для визначення поведінки неігрових персонажів, він залишається складним та схильним до помилок процесом, що вимагає високого рівня технічних знань.

Майже в кожній комп’ютерній грі є певна кількість персонажів, які зустрічаються гравцеві по ходу проходження гри. У кожного персонажу комп’ютерної гри є своє призначення, яке визначається контекстом та правилами

гри. Особливими персонажами, які стоять осторонь, є персонажі, які називаються NPC. Саме скорочення походить від англійського NPC, яке розшифровується як Non-Playable-Character. Це неігровий персонаж, який зустрічається в грі, якого гравець не може взяти під свій контроль, тому що ним керує комп'ютер. Іншими словами це штучний інтелект, тобто бот, що має форму персонажу гри.

В комп'ютерних іграх терміном NPC позначаються персонажі, що спілкуються з гравцем незалежно від їх відношення до ігрового персонажу. Вони можуть бути дружніми, нейтральними або ворогуючими. Неігрові персонажі є важливим засобом створення атмосфери гри, слугують мотивуючим чинником для гравців здійснювати ті або інші дії і є основним джерелом інформації про світ гри та сюжет гри.

Цей персонаж має власну запрограмовану систему дій якої він дотримується. Раніше у NPC було мало реплік та ігрових дій. Наразі, в сучасних іграх, їх можливості розширилися і в багатьох іграх у персонажів NPC наявні великі програми з можливістю прийняття власних рішень та різними реакціями на ті чи інші дії гравця. Це досягається за рахунок використання машинного навчання. Машинне навчання допомагає вирішувати такі типи завдань як розпізнавання та опрацювання великих масивів даних, класифікація та аналіз цих даних, побудова рішень з прогнозами та інше. Ці можливості машинного навчання використовуються для розширення та спрощення елементів ігрового дизайну. Штучний інтелект та машинне навчання вже стали важливою частиною розробки комп'ютерних ігор завдяки спрощення процедурної генерації та оптимізації існуючих алгоритмів. Разом з тим, існують певні недоліки, як то тривалий час навчання певних моделей або велика кількість даних для розвитку гри. Результат, як правило, залежить від якості та кількості вхідних даних на основі яких проводиться навчання. Некоректність, недостатність або хибність вхідних даних призводить до того, що результат роботи моделі машинного навчання може бути неочікуваним або хибним.

На сьогоднішній день багато дослідників вивчають питання застосування машинного навчання для персонажів комп'ютерних ігор. Зокрема, автори в роботі

[43] описують метод, що використовує не детерміноване планування поведінки ігрових персонажів для створення різноманітних політик рішень, що можуть виконуватись як кінцеві автомати, що керують поведінкою NPC.

В роботі [44] пропонується підхід, що суміщає застосування ідей недетермінованого евристичного пошуку та використання запропонованої авторами евристики T-Graph, що дозволяє гравцям та розробникам навчати неігрових персонажів новим моделям поведінки, що не є жорстоко запрограмованими. У відеоіграх генерація поведінки неігрового персонажу звичайно залежить від жорсткого кодування дій NPC. Але в багатьох ігрових ситуаціях важко передбачити як NPC повинен себе поводити. Автори дослідили підходи до планування на основі пошуку, використовуючи демонстрацію для управління, пошук в багатовимірних просторах, що представляють собою повний стан гри. Вони розробили евристику Training Graph, розширення евристики Experience Graph, котра направляє пошук плавно та ефективно, навіть тоді, коли демонстрація недоступна у просторі пошуку і гарантує, що більша кількість демонстрацій використовується для навчання поведінки неігрових персонажів. На високому рівні розроблений планувальник починається з початкового стану в графі, що надає доступні стани простору пошуку по ребрах, які їх пов’язують. Алгоритм пошуку знаходить шлях через цей граф для досягнення цільового стану. Робота може бути використана як базова для створення навчаємого штучного інтелекту неігрових персонажів, що дозволить гравцям у відеоігри використовувати нову механіку ігрового процесу та допоможе покращити відео.

Автори роботи [45] представили узагальнений метод інтегрування перетворення агентних моделей у серйозні ігри та стимулятори. У запропонованій парадигмі людина-гравець бере на себе роль одного агента в моделі, в той час як вихідні дані моделі контролюють середовище, правила взаємодії агентів та всіх інших агентів (неігрових персонажів) з якоми людина-гравець взаємодіє. Крім того, створено прототипи двох систем та представлено два сценарії використання, що демонструють як використання агентних моделей в якості

контролерів поведінки для неігрових персонажів вносить деяку непередбачуваність у віртуальну стимуляцію.

Дослідження [46] розглядає навчання на демонстраціях. Це є перспективна галузь, яка вивчає, як створити інтелектуальних агентів, здатних відтворювати поведінку, навчаючись на демонстраціях людських експертів. У попередній роботі авторів агент повністю контролював процес навчання, тому вирішував, коли відмовитись або відновити контроль над персонажем. У цій роботі представлено агента для обґрунтування в Інтернеті та на основі кейсів, який навчається імітувати реальних гравців Pac-Man за допомогою інтерактивного підходу, в якому як людський гравець, так і обчислювальний агент по черзі керують головним героєм. Систему вдосконалено, щоб гравець також міг відновити контроль над персонажем і повернутися в минуле, щоб виправити неправильну поведінку, що виявляється агентом, щоразу, коли вони виявляються. Автори також представили оцінку системи, проведену трьома професійними дизайнерами відеоігор.

Автори дослідження [47] розглянули глибоке навчання з підкріпленням навчальних агентів складної поведінки у віртуальних 3D-середовищах на прикладі Майнкрафт. В роботі проаналізоване інтерактивне машинне навчання, коли вчителі-люди відіграють безпосередню роль у навчанні за допомогою демонстрацій, критики чи порад щодо дій, що, в свою чергу, може полегшити сприйнятливість агента до псевдонімів. Однак інтерактивне машинне навчання практичне лише тоді, коли кількість людських взаємодій обмежена, що вимагає балансу між зусиллями вчителя та ефективністю роботи агента. Було проведено експерименти з двома додатковими алгоритмами навчання, які дозволяють вчителям-людям давати поради щодо дії. Цими алгоритмами є арбітраж зворотного зв’язку (Feedback Arbitration) та поради щодо дії Ньютона (Newtonian Action Advice) – у візуальних умовах згладжування. Алгоритм Newtonian Action Advice [Krening 2018] побудований на табличному Q-навчанні, де для кожного стану та дії зберігається таблиця q-значень. Кренінг та інші (2018) виявив, що люди-тренери сприймають агент Ньютоновських дій як більш розумний,

ефективніший, прозоріший ніж стандартний агент Q-навчання. Автори дійшли висновку, що Newtonian Action Advice (NAA) покращує ефективність навчання агента всередині областей з високим перцептивним псевдонімом. Агент NAA в порівнянні з агентом Feedback Arbitration не тільки притримується політики, яка дозволяє швидше отримати високу нагороду, але й якісно витрачає більшу частину свого навчального часу, рухаючись по областях високого сприйняття згладжування. Це досягається, незважаючи на те, що як і FA, так і NAA отримують ті самі поради, з тією ж точністю прогнозів і з однаковою частотою порад.

В роботі [48] представлене навчання з підкріпленнім для неігрового персонажу. Навчання з підкріпленням – це алгоритм самонавчання, в якому агент набуває досвіду завдяки постійній взаємодії з навколишнім середовищем, що більше нагадує процес навчання людей або тварин. Особливістю навчання з підкріпленням є те, що класичний алгоритм навчання з підкріпленням може легко створити так зване прокляття розмірності, коли розмір стану занадто великий. З метою покращення коефіцієнта конвергенції навчання з підкріпленням пропонується метод тренування неігрового персонажу в іграх із використанням алгоритму навчання Сарса. Штучна нейронна мережа використовується для наближення функції значення. Для того, щоб краще використовувати набутий досвід, в роботі пропонується створення подвійної нейронної мережі та використання пам’яті досвіду для зберігання досвіду, а також використовується перегляд досвіду, щоб пришвидшити збіжність алгоритму Сарса. Доведено, що використовуючи метод, представлений у цій роботі для навчання NPC є можливість знайти NPC, який навчається за допомогою цього методу, і має більше можливостей до навчання, ніж класичне навчання з підкріпленням.

Автори роботи [49] пропонують створювати поведінки неігрових персонажів інтерактивно навчаючи агента в цільовому середовищі, використовуючи імітаційне навчання з людиною. Наразі існує великий попит на високоякісні персонажі гравців NPC у відеоіграх. Прописувати та керувати їх поведінкою вручну для дизайнерів гри є трудомістким і схильним до помилок

інженерним процесом з обмеженим контролем. Автори пропонують створювати такі поведінки NPC інтерактивно навчаючи агента в цільовому середовищі, використовуючи імітаційне навчання з людиною. Хоча традиційне клонування поведінки може бути недостатнім для досягнення бажаної продуктивності, автори доводять в роботі, що інтерактивність можна суттєво покращити за допомогою незначних людських зусиль. Модель тренується з використанням ансамблю Маркова з великою роздільною здатністю моделі, які можна використовувати як в такому вигляді, так і надалі стиснуту в більш компактну модель в користувацьких пристроях. Цей підхід проілюстрований на прикладі OpenAI Gym, де людина може швидко допомогти навчити агента лише за кілька інтерактивних демонстрацій. В цій роботі показано, що включення людини до навчання може призвести до фективного практичного навчання, навіть такокої простої моделі, як модель ансамблю Маркова. Квантування забезпечує простий, але водночас ефективний спосіб узагальнення для моделей, що навчаються лише на обмеженій кількості людських даних.

В робота [50] розглядається гра з жанром. Рольова гра (RPG) - це змагальна гра між гравцями проти інших гравців або ворогів у формі персонажа, що не є гравцем (NPC). Багато розробників ігор, роблячи саму гру, як і раніше використовують ручні методи для визначення атрибутів ігрового процесу для персонажів гравця або ворога. У цьому дослідженні в програмі використано кілька підходів, таких як k-NN, Normal Distribution та Naive Bayes, для автоматичного розрахунку зростання атрибутів ігрового процесу. Розподіл атрибутів гравця виконується двічі з однаковими входами, щоб знайти максимальний рівень рівня кожного збільшення атрибута. Це доводить, що атрибут збільшується і розподіляється по рівнях є відповідниками вхідної змінної. Вороги перебувають у звичайному розподілі з найбільшою ймовірністю появи між рівнем 0 і 100. Для типів ворогів вхідні змінні для розподілу ворогів за типом відповідають результату.

В роботі [51] автори розширюють поняття дерева поведінки. Це метод створення поведінки, що є популярним у відео ігровій індустрії, з трьома новими

типами вузлів, що полегшує розробку та реалізацію неігрових персонажів, які повинні узгоджуватись між собою. Автори пропонують реалізацію та методологію для належного використання вузлів координації нашого розширення та показують як ними користуватися з розробленим сценарієм застосування. В останні роки в наукових дослідженнях як теоретичних, так і практичних координація мультиагентних систем була досить сильною. Щось подібне відбувається з розробкою нових інструментів для індустрії відеоігор. Авторський підхід сприяє обом напрямкам надаючи нове розширення, яке полегшує розробку та реалізацію агентів, які повинні координувати між собою. У відеогірах агенти або NPC персонажі контролюються не гравцем, а грою за допомогою алгоритмічної, заздалегідь визначеної поведінки чи реагування, або за допомогою більш досконалої техніки штучного інтелекту. Для деяких відеоігор потрібні NPC з динамічними, надійними та розумно непередбачуваної поведінкою, щоб тримати гравців зайнятими та зануреними в гру. Замість того, щоб наділяти NPC з дуже складною поведінкою особистості, здійснений сосіб поліпшити їх непередбачуваність в розумний та надійний спосіб, що дозволяє їм координувати між собою. Оскільки дерева поведінки зосереджені на створенні індивідуальної поведінки, добитися скоординованої поведінки можливо шляхом жорсткого кодування самої координації. Але це спеціальне рішення частково знижує деякі переваги, які популяризують дерева поведінки: бути візуально інтуїтивно зрозумілим, масштабованим та багаторазовим. З цієї причини автори пропонують розширення дерев поведінки, яке розробники можуть використовувати NPC, не йдучи проти парадигми розвитку: створення складної поведінки шляхом проєктування інтуїтивно зрозумілої деревної структури.

В роботі [52] автори в ході експерименту дослідили чотири підходи до штучного інтелекту NPC. Ці підходи стосуються використання навчання з підкріпленням Q-learning, генетичні алгоритми, дерево рішень та гібридний підхід, що об’єднує в собі генетичні алгоритми та Q-learning. Система прийняття рішень була змодельована як багатошаровий персептрон нейронної мережі з одним шаром прихованих нейронів. Мережа використовує пряму техніку обробки

входів від датчиків агента NPC. Датчики подаються мережею з такою інформацією, як: де знаходиться NPC – в повітрі/ на землі; може NPC побачити ворога; яким шляхом і як далеко ворог; якою була остання дія NPC і т.д. Вихід нейронної мережі – це наступна дія NPC, яка вибирається із заданого набору (поворот ліворуч/праворуч, рух вперед, атака, стрибок).

Для порівняння підходів були використані наступні метрики:

1. Швидкість навчання NPC в кожному поколінні.

2. Значення функції фітнесу для кожного покоління NPC.

3. Сумарна квадратична похибка кожного NPC в кожному поколінні.

Персонажі, що навчались методом Q-learning не мали можливість передбачати рахунки інших персонажів в грі. Нейронна мережа змогла імітувати функцію Q навіть після 20 поколінь особин (що триває 400 секунд при прискореному моделюванні). Це було прийнятним до надмірного (або перенавченого явища), що призвело до збільшення похибки квадрата через деякий час. Перенапруження спричинене занадто точним регулюванням моделі для вивчення даних. У цьому випадку реальні дані навчання створюються в режимі реального часу з методом Q-learning. Чим більше переповнена модель, тим більше шкоди завдає мутація ваги одиночного нейрону. Основним вузьким місцем тут є значення ваг, воно коливається близько нуля і навіть незначна зміна (вгору або вниз) може призвести до швидких змін, що пов’язані з класифікацією штучних нейронних мереж. Надмірно перенавчені індивідуальні особи характеризуються меншою пристосованістю та сприйнятливістю до другорядних змін, наприклад генетичні мутації чи зміна остаточної функції. Був використаний метод ранньої зупинки, щоб уникнути збільшення моделі. Метод створював резервні копії нейронної мережі відразу після того, як спостерігалось значне збільшення загальної квадратичної помилки. Надмірість не завжди сприяиняє помітне зниження якості поведінки агентів, іноді генетичний алгоритм створював рішення, які були визнані Q-learning набагато гіршим, хоча для людини спостерігача вони працювали дуже добре, якщо аналізувати значення фітнес-функції. З метою запобігання обмеження еволюційного потенціалу було прийнято

рішення відключити зворотне поширення ваги реалізоване в алгоритмі Q-learning одразу після досягнення низьких та стабільних значень загальної квадратичної похибки. Генетичний алгоритм в такому випадку заміщений Q-learning, ще більше збільшує фізичну працездатність людей. Діаграми функцій фітнесу для навчання мереж за допомогою техніки Q-learning характеризувались загальним збільшенням коливань навколо логарифмічної кривої.

Було проведено порівняння двох методів використання Q-learning та генетичного алгоритму, а потім був проаналізований гібридний метод з двох компонентів цих методів. Результати проведеного аналізу представлені в таблиці 2.1.

Таблиця 2.1 – Таблиця порівняльного аналізу методів

Метрики	Метод Q-learning	Генетичний алгоритм (GA)	Комбінований метод (QL+GA)
Швидкість навчання нейронної мережі	Зупинено при досягненні 165 поколінь	Зупинено при досягненні 230 поколінь	Зупинено при досягненні 200 поколінь
Максимальне значення фітнес функції	Подібне	Подібне	Подібне
Сумарна квадратична похибка	25	40	32
Продуктивність	Низька	Зросла	Найкращі результати багатошаровий персептрон з правилами QL+GA

Першим і найбільш очевидним фактором порівняння була швидкість навчання нейронної мережі. Тоді максимальні значення фітнес функції досліджували як вторинні. Кожна вихідна популяція мала ваги, вибрані випадковим чином з одного і того ж набору.

Кожна вихідна популяція мала ваги, що були вибрані навмання з однієї і тієї ж вибірки і окремі люди були готові приймати рішення детермінованим чином (перемагаючи вихідний нейрон передбачає остаточну дію). Це дозволило уникнути випалковості і зробити висновки для порівняння. Швидкість побудови моделей ШІ за допомогою QL була помітно більшою, ніж за допомогою постійного потоку даних, що надходили з навколишнього середовища. Ізольований підхід GA потребує в кілька разів більше поколінь для досягнення середніх значень фітнес функції порівняно з результатами QL. Щоб запобігти перенавчанню моделі використовувався метод ранньої зупинки. Цей метод створював резервні копії мережі одразу після спостереження значного збільшення загальної квадратичної похибки. Не завжди перенавчання приводило до помітного падіння якості в поведінці агентів – деколи генетичні алгоритми створювали рішення, які були визнанні QL набагато гіршими, хоча для людини-спостерігача і аналізуючи значення фітнес-функції вони працювали гарно. Щоб не допустити обмеження еволюційного потенціалу, було прийнято рішення виключати обернене розповсюдження ваг реалізоване в алгоритмі QL одразу після досягнення низьких та стабільних значень сумарної квадратичної похибки. З поколінь коли модель вже навчилась імітувати дерево рішень, спостерігаються коливання між двома рівнями, які є результатом перенавчання мережі. Кожен підвищений рівень викликаний необхідними випадковими мутаціями ваги нейронів. Результати показують, що максимальні значення фітнес-функції особин в обох методах були подібними, але підхід з генетичним алгоритмом до досягнутих значень повинен асоціюватись з коефіцієнтом з великою випадковістю. Використання генетичного алгоритму не спричиняє проблем з продуктивністю. Але коли використовується машинне навчання з великою кількістю рекурсивних кроків, то виникають проблеми. Кількість повторних кроків у функції якості мали великий вплив на результативність процесу навчання. В результаті відбулося подальше збільшення глибини рекурсії з великою затримкою, що заважає годиннику моделювання та відключає потенціал геймплей.

В результаті порівняння популярних методів, що впливають на інтелектуальну поведінку представлені відтворювані персонажі відеоігор. Результати показують, що найкращі результати можуть бути досягнуті за допомогою гібридного методу, що складається з багатошарового персепторну з правилами генетичного алгоритму та машинного навчання з підкріпленням.

2.3 Агенто-орієнтований метод інтелектуалізованої системи

Метою даної роботи є створення штучного інтелекту NPC комп'ютерної гри, який задовільнить як функціональну частину гри, так і дизайнерьску. За основу було обрано навчання з підкріпленням, де агент (NPC) буде обирати та здійснювати дії, що б отримати максимальний бал. З більш відомих ігор, які схожі переслідували схожу мету була – Facade 2005 року [53]. Це проста інді-гра на 15 хв. Головною особливістю цієї гри є просунутий штучний інтелект персонажів. У цій грі сюжет не має однозначності, існує декілька варіантів розвитку подій та три типи закінчення гри. При наступній грі сцени вже можуть бути абсолютно іншими і сценарій відповідно змінюється кардинальним чином. По сюжету гравець приходив в гості до знайомої пари, з управління він лише міг пересуватися і комунікувати з сім'єю за допомогою вводу тексту (а не заздалегідь заготовлених відповідей), що було на той момент досить революційно. Завдяки використанню штучної нейронної мережі, ігрові персонажі давали більш-менш адекватну відповідь на повідомлення гравця. Коли ця гра тільки вийшла, багато хто пророкував, що це прорив і початок нової ери штучного інтелекту в іграх, але дива не сталося, розробники ще не зовсім вміли працювати з новою (для них) технологією, машинне навчання вело себе непередбачувано і могло сильно зіпсувати ігровий досвід, тому від нього відмовилися.

В даній дипломній роботі розглядається наступна ключова механіка комп'ютерної гри. Це кулачні бої. Якщо опустити засоби переміщення і орієнтації, то у NPC буде доступно 8 дій і 8 збудників від яких буде прийнято рішення про наступні ходи:

1. Nothing – нічого не відбувається.
2. Left Attack – атака лівою.
3. Right Attack – атака правою.
4. Left Block – лівий блок.
5. Right Block – правий блок.
6. Left Fint – фінт лівою.
7. Right Fint – фінт правою.
8. Grab – захоплення (схопити/вирватись).

Штучний інтелект змінює стан у двох наступних випадках:

1) Коли гравець виконав дію та штучний інтелект було про це повідомлено;

2) Кожні 2 секунди.

Штучний інтелект перевіряє отриманий стан (наприклад, гравець виконав удар лівою рукою), звертається до таблиці (таблиця 2.2) і обирає оптимальну дію. Після чого застосовує обрану дію, після її завершення отримує результат та оновлює таблицю.

Таблиця 2.2 – Таблиця початкових станів та дій штучного інтелекту гри

ID		Нічого	Удар лівою	Удар правою	Блок лівою	Блок правою	Фінт лівою	Фінт правою	Вирватись
		1	2	3	4	5	6	7	8
1	Нічого	0	0	0	0	0	0	0	0
2	Удар лівою	0	0	0	0	0	0	0	0
3	Удар правою	0	0	0	0	0	0	0	0
4	Блок лівою	0	0	0	0	0	0	0	0
5	Блок правою	0	0	0	0	0	0	0	0
6	Фінт лівою	0	0	0	0	0	0	0	0

Кінець таблиці 2.2 – Таблиця початкових станів та дій штучного інтелекту гри

ID	Нічого	Удар лівою	Удар правою	Блок лівою	Блок правою	Фінт лівою	Фінт правою	Вирватись
	1	2	3	4	5	6	7	8
7	Фінт правою	0	0	0	0	0	0	0
8	Схопити	0	0	0	0	0	0	0

Для моделювання дій штучного інтелекту використовується скінченний автомат. В даній роботі скінченний автомат використовується для створення дерева рішень штучного інтелекту. Головними ознаками скінченного автомату являються наступні ознаки:

- наявність фіксованого набору станів, який притаманний даній моделі;
- модель може перебувати одночасно виключно в одному стані;
- кожна модель отримує свою послідовність вхідних даних;
- кожен стан моделі має певний набір переходів, в свою чергу кожен перехід пов'язаний з набором вхідних даних і вказує на певний стан.

Використання скінченного автомату дозволяє покращити штучний інтелект персонажів гри. Поведінка персонажу складається з восьми компонент в залежності від ігрової ситуації. Введення до архітектури гри спеціального модуля, що відповідає за скінченний автомат дає можливість задавати множину станів ігрового персонажу, умови переходу з стану в стан та можливість вибору стану по пріоритету в залежності від ігрової ситуації при наявності декількох можливих варіантів.

Агентна модель навчання з підкріпленням

$$AGP = (Zs, St, Rt, Ag, Vg, Wr), \quad (2.1)$$

де *Zs* – зовнішнє середовище;

St – стан;

Rt – винагорода;

Ag – набір дій агента;

Vg – набір дій суперника;

Wr – ваговий коефіцієнт винагороди.

Агент взаємодіє з зовнішнім середовищем, переходить з стану в стан та отримує винагороду за правильно виконану дію. В нашій роботі агент змінює свій стан в залежності від впливу зовнішнього середовища. Множина станів агента представлена формулою 2.2.

$$St = (st_0, st_1, st_2, st_3, st_4, st_5, st_6, st_7), \quad (2.2)$$

де st_0 – початковий стан агента;

$st_1 \ldots st_7$ – стани агента в залежності від впливу зовнішнього середовища.

Множина вагових коефіцієнтів винагороди агента представлена формулою 2.3.

$$Wr = (wr_1, wr_2, wr_3, wr_4, wr_5), \quad (2.3)$$

Вагових коефіцієнтів є п’ять, розподіляються вони наступним чином. Відбуваються певні дії, в залежності від зовнішнього середовища, за які агент отримує винагороду.

На вхід подається результат дії, якщо нічого не відбулось – функція повертає – 0 очок, якщо агент отримав шкоди, то функція повертає -25 очок, якщо ж агент наніс шкоди, то 25 очок, при блокуванні удару повертається 5 очок і відповідно, якщо агент вдарив по блоку – -5 очок.

Значення в таблиці змінюється за формулами 2.4 - 2.6.

$$Q(S_t, a) = R_t + y\, max_a Q(S_{t+1}, a), \quad (2.4)$$

$$\Delta = R_t + y\, max_a Q(S_{t+1,} a) - Q(S_t, a), \quad (2.5)$$

$$Q(S_{t+1}, a) = Q(S_t, a) + a\Delta, \quad (2.6)$$

Це свідчить про те, що якість пари дія- стан є безпосередньою винагородою плюс знижена вартість усіх успішних станів, що мають відповідний ваговий коефіцієнт. Значення стану становить якість найкращих дій для цього стану. Звідси випливає, що знання *Q* достатньо, щоб визначити оптимальну стратегію для кожного стану, і ми можемо вибрати дію з найвищим *Q*-значенням. Враховуючи, що це ітераційний процес, то він починається з оцінок для *Q* та *Rt*, генерує нові оцінки.

Для прийняття рішень агентом використовується процес Маркова. Фрагмент взаємодії станів агента представлений на рисунку 2.1.

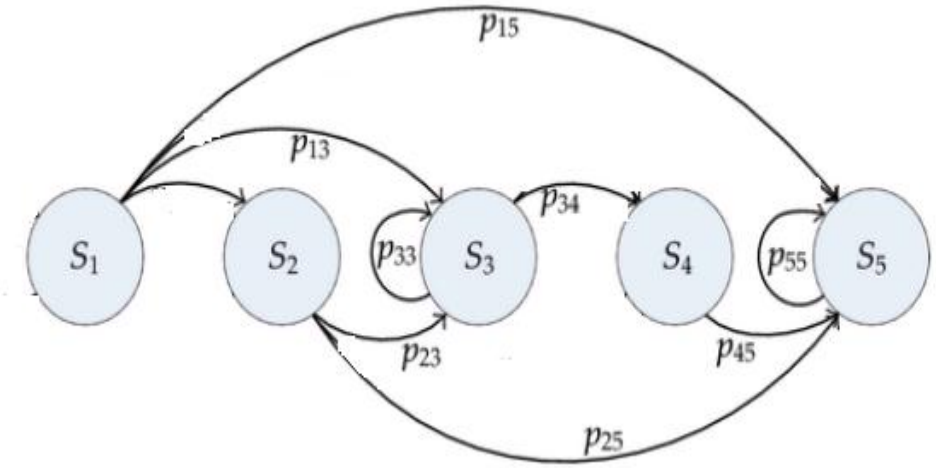


Рисунок 2.1 – Фрагмент взаємодії станів агента

У процесі прийняття рішень Маркова формалізація навчання з підкріплення є єдиною адаптацією. В даному випадку агент взаємодіє із середовищем, визначеним за допомогою використання імовірнісної перехідної функції. З цієї точки зору, вторинні агенти можуть бути лише частиною оточення і тому

фіксуються у своїй поведінці. Фреймворк ігор Маркова дозволяє розширити його застосування, включивши безліч адаптивних агентів із взаємодіючими або конкуруючими цілями. За допомогою процеса Маркова описується Q-Learning-подібний алгоритм для пошуку оптимальної агентної стратегії.

У цьому формулюванні Q-навчання оновлення виконує агент, коли він отримує винагороду, коли робить перехід від стану в стан після певної дії. При отриманні винагороди обов'язково враховується ваговий коефіцієнт цієї винагороди. Ймовірність з якою це відбувається є такою, що для агента можливо провести відповідне оновлення без явного використання ймовірнісного підходу. Це правило навчання сходиться до правильних значень для *Q і Rt*, припускаючи, що кожна дія випробовується в кожному стані нескінченно часто і що нові оцінки поєднуються з попередньо використаними досить повільного експоненціально зваженого середнього.

Змінними від середовища є: набір станів, St; набір дій агента, Ag; набір дій суперника Vg . Внутрішніми для агента змінними є: швидкість навчання, альфа, яка ініціалізується до 1,0 і занепадає через деякий час; оцінка агентом Q-функції; оцінка агентом Vg-функції; вагові коефіцієнти винагороди і поточну політику агента щодо станів. Решта змінних є параметрами алгоритму: змінна, що контролює, як часто агент відхиляється від своєї поточної стратегії, щоб забезпечити адекватне дослідження та контролює швидкість, з якою швидкість навчання знижується. Цей алгоритм називається мінімакс-Q, оскільки він є по суті ідентичний стандартному алгоритму Q-навчання з мінімаксом, що замінює макс.

Агентно-орієнтований метод побудований на основі моделі агента з використанням врахування вагових коефіцієнтів винагороди та алгоритмів машинного навчання. Основні кроки агентно-орієнтованого методу представлені на рисунку 2.2.

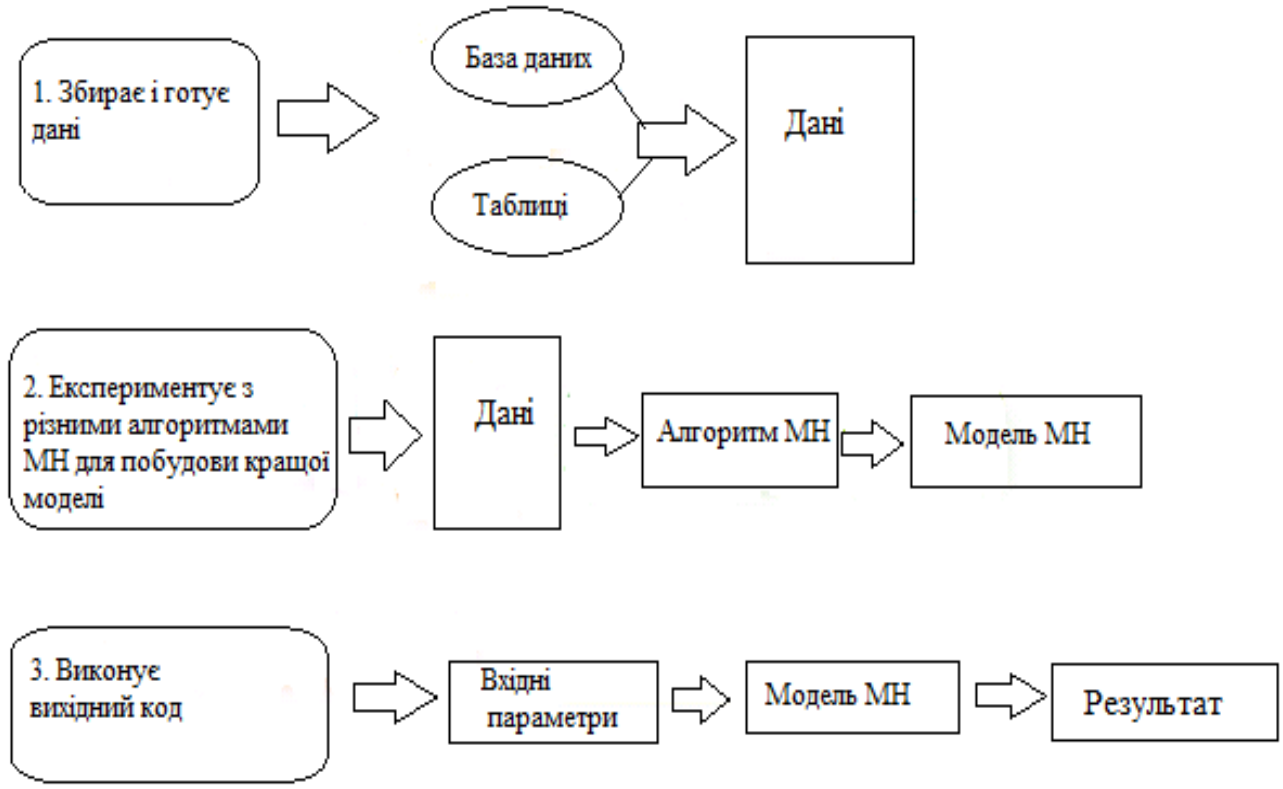


Рисунок 2.2 – Основні кроки агентно-орієнтованого методу

На першому кроці метод збирає та готує дані для опрацювання, використовуючи іформацію з баз даних та таблиць, де міститься інформація про зовнішнє середовище, самого агента, його стани, та реакції на зміну станів. На другому кроці метод експериментує з різними алгоритмами машинного навчання для того, щоб мати змогу побудувати кращу модель. Алгоритми машинного навчання застосуваються до даних, які були зібрані на попередньому кроці. В якості результатів, після застосування різних алгоритмів та вибору з них найкращого, формується модель машинного навчання, яка і буде працювати з агентом та зовнішнім середовищем. Третій і кінцевий крок методу призначений для виконання вихідного коду. При цьому аналізуються вхідні параметри, отримані на попередніх кроках, будується модель машинного навчання та отримується результат у вигляді оптимальної агентної стратегії.

2.4 Висновки

У другому розділі роботи представлена формалізація процесу прийняття рішення з використанням ланцюгів Маркова та модель агента, а також представлений удосконаленний агентно-орієнтований метод, що дозволяє підвищити продуктивність агента за рахунок врахування вагового коефіцієнта винагороди, що в свою чергу задовільняє функціональну та дизайнерьску систему гри. Проведено аналіз існуючих алгоритмів навчання та обгрунтовано переваги методу машинного навчання з підкріпленням, що доцільно використовувати для навчання агентів у системі. Також досліджено проблеми, які виникають у процесі навчання та способи їх усунення.

3 МЕТОДИ ТА АЛГОРИТМИ ІНТЕЛЕКТУАЛІЗОВАНОЇ СИСТЕМИ МАШИННОГО НАВЧАННЯ

3.1 Використання мікрокомп’ютерів для машинного навчання

У тих випадках, коли мова йде про машинне навчання з використанням мікрокомп’ютерів, то зауважимо, що модель навчання з підкріпленням не може навчатись на RaspberryPi або з використанням іншого мікрокомп’ютера тому, що на їх платах не вистачає потужностей для того, щоб виконувати велику кількість операцій. Під час навчання велика кількість операцій множення з плаваючою комою займає всі обчислювальні ресурси мікрокомп’ютера, отже є можливість тільки імпортувати та запускати вже готову навчену модель на мікрокомп’ютерах.

Розглянемо основні плати мікрокомп’ютерів, їх параметри і можливості додаткових модулів для машинного навчання. В роботі [54] наведений огляд порівняння ефективності мікрокомпютерних платформ машинного навчання. За результатами проведеного огляду зроблено висновок, що для навчання найефективнішим рішенням за доступною ціною буде комбінація мікрокомп’ютера Raspberry Pi 4 з Intel Neural Compute Stick 2 і фреймворком TensorFlow. Порівняння ефективності мікрокомп’ютерних платформ машинного навчання наведені в таблиці 3.1.

TensorFlow від Google – відкрита й найпопулярніша бібліотека з глибокого навчання для досліджень і розроблення. TensorFlow - це наскрізна платформа, яка спрощує створення та розгортання моделей ML. TensorFlow пропонує кілька рівнів абстракції, щоб можна було вибрати відповідний для своїх потреб. Можна створювати та навчати моделі за допомогою високорівневого API Keras, що полегшує початок роботи з TensorFlow та машинним навчанням. Якщо потрібна більша гнучкість, наполегливе виконання дозволяє негайну ітерацію та інтуїтивне налагодження.

Для великих навчальних завдань машинного навчання використовуються API стратегії розподілу для розподіленого навчання на різних апаратних конфігураціях без зміни визначення моделі.

Таблиця 3.1 – Порівняння ефективності мікрокомп’ютерних платформ машинного навчання [Могильний С. Б.]

Модель	Фреймворк	Raspberry PI (use TF-Lite)	Raspberry PI (our NCNN)	Raspberry PI Intel Neural Stick 2	Raspberry PI Google Coral USB	JeVois	Jetson Nano	Google Coral
EfficientNet-B0 (224x224)	TensorFlow	14,6 FPS (PI 3) 25,8 FPS (PI 4)	-	95 FPS (PI 3) 180 FPS (PI 4)	105 FPS (PI 3) 200 FPS (PI 4)	-	216 FPS	200FPS
ResNet-50 (244x244)	TensorFlow	2,4 FPS (PI 3) 4,3 FPS (PI 4)	1,7 FPS (PI 3) 3 FPS (PI 4)	16 FPS (PI 3) 60 FPS (PI 4)	10 FPS (PI 3) 18,8 FPS (PI 4)	-	36 FPS	18,8 FPS
MobileNet-v2 (300x300)	TensorFlow	4,4 FPS (PI 3) 8 FPS (PI 4)	8 FPS (PI 3) 8,9 FPS (PI 4)	30 FPS (PI 3)	46 FPS (PI 3)	30 FPS	64 FPS	130 FPS
SSD MobileNet-v2 (300x300)	TensorFlow	2,6 FPS (PI 3) 4,7 FPS (PI 4)	3,7 FPS (PI 3) 5,8 FPS (PI 4)	11 FPS (PI 3) 41 FPS (PI 4)	17 FPS (PI 3) 55 FPS (PI 4)	-	39 FPS	48 FPS
Binary model (300x300)	XNOR	6,8 FPS (PI 3) 12,5 FPS (PI 4)	-	-	-	-	-	-
Inception V4 (299x299)	PyTorch	-	-	-	3 FPS (PI 3)	-	11 FPS	9 FPS
Tiny YOLO V3 (416x416)	Darknet	0,5 FPS (PI 3) 1 FPS (PI 4)	1,1 FPS (PI 3) 1,9 FPS (PI 4)	-	-	2,2 FPS	25 FPS	-
OpenPose (256x256)	Caffe	-	-	5 FPS (PI 3)	-	-	14 FPS (PI 4)	-
Super Resolution (481x321)	PyTorch	-	-	0,6 FPS (PI 3)	-	-	15 FPS	-

Кінець таблиці 3.1 – Порівняння ефективності мікрокомп’ютерних платформ машинного навчання [Могильний С. Б.]

Модель	Фреймворк	Raspberry PI (use TF-Lite)	Raspberry PI (our NCNN)	Raspberry PI Intel Neural Stick 2	Raspberry PI Google Coral USB	JeVois	Jetson Nano	Google Coral
VGG-19 (224x224)	MXNet	0,5 FPS (PI 3) 1 FPS (PI 4)	-	5 FPS	-	-	10 FPS (PI 4)	-
Unet (1x512x512)	Caffe	-	-	5 FPS	-	-	18 FPS	-

Більша частина обчислень, що відповідає за штучний інтелект передається визначеному блоку, який розробляється під ці потреби, на процесорі, завдяки цьому підвищується продуктивність самого додатку в декілька разів.

Перспективні алгоритми, інструменти та платформи для машинного навчання представлені на рисунку 3.2.

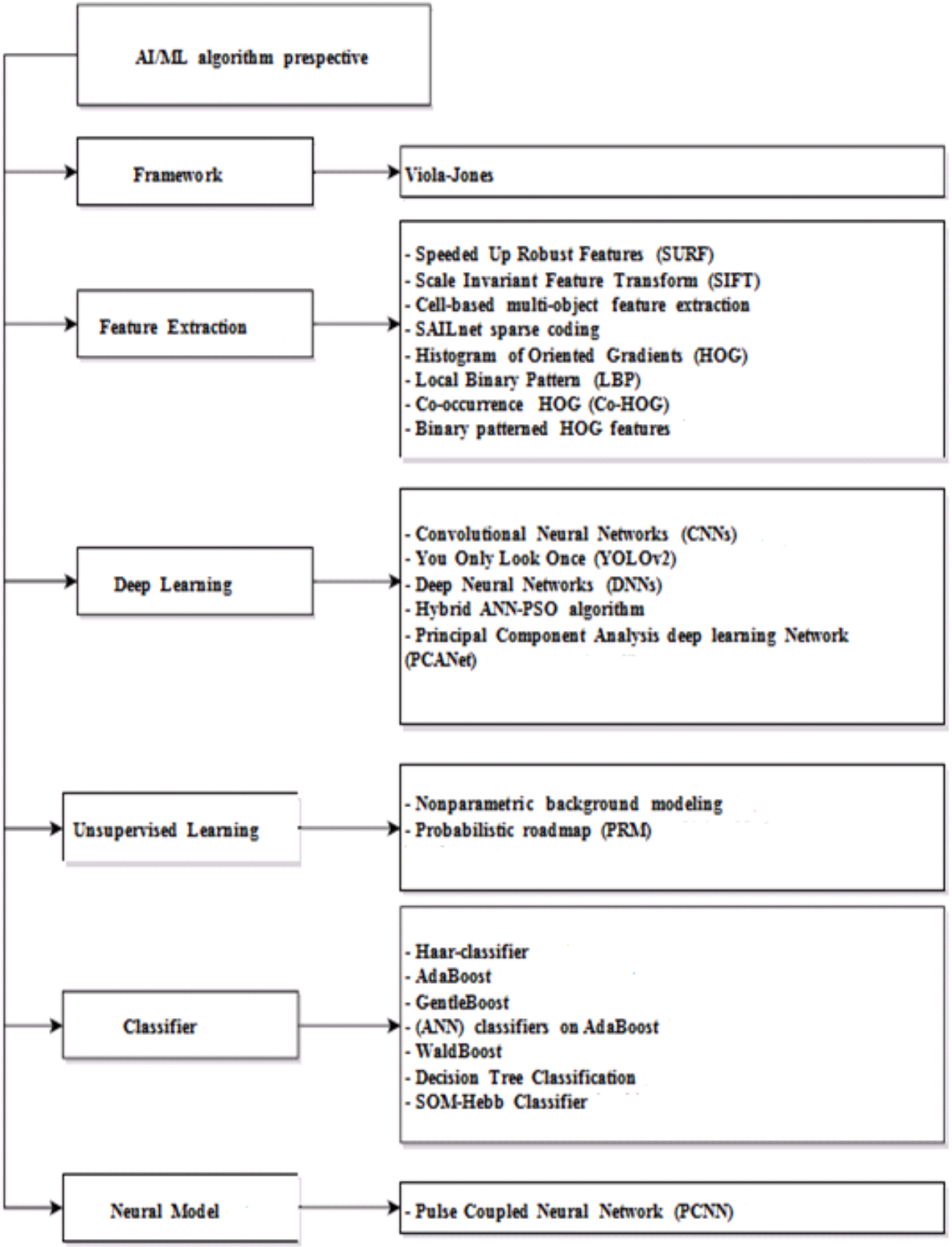


Рисунок 3.2 – Перспективні алгоритми, інструменти та платформи для машинного навчання

Важливим питанням при обрахунку штучного інтелекту в іграх та застосуванні машинного навчання є прискорення, що дозволяє ефективно вирішити питання розподілу ресурсів процесора та зменшення часу навчання штучного інтелекту. Розглянемо існуючи та найбільш часто використовувані апаратні прискорювачі штучного інтелекту та машинного навчання, засновані на стандартних показниках якості. Документи апаратних прискорювачів класифікуються з апаратної точки зору, щоб забезпечити корисне порівняння. Це п’ять категорій, перелічених нижче та представлених на рисунку 3.3:

- апаратне картографування;
- дизайн прискорювача на основі FPGA;
- дизайн прискорювача на основі графічного процесора;
- перекласифікація прискорювачів;
- дизайн прискорювача Xeon-Phi.

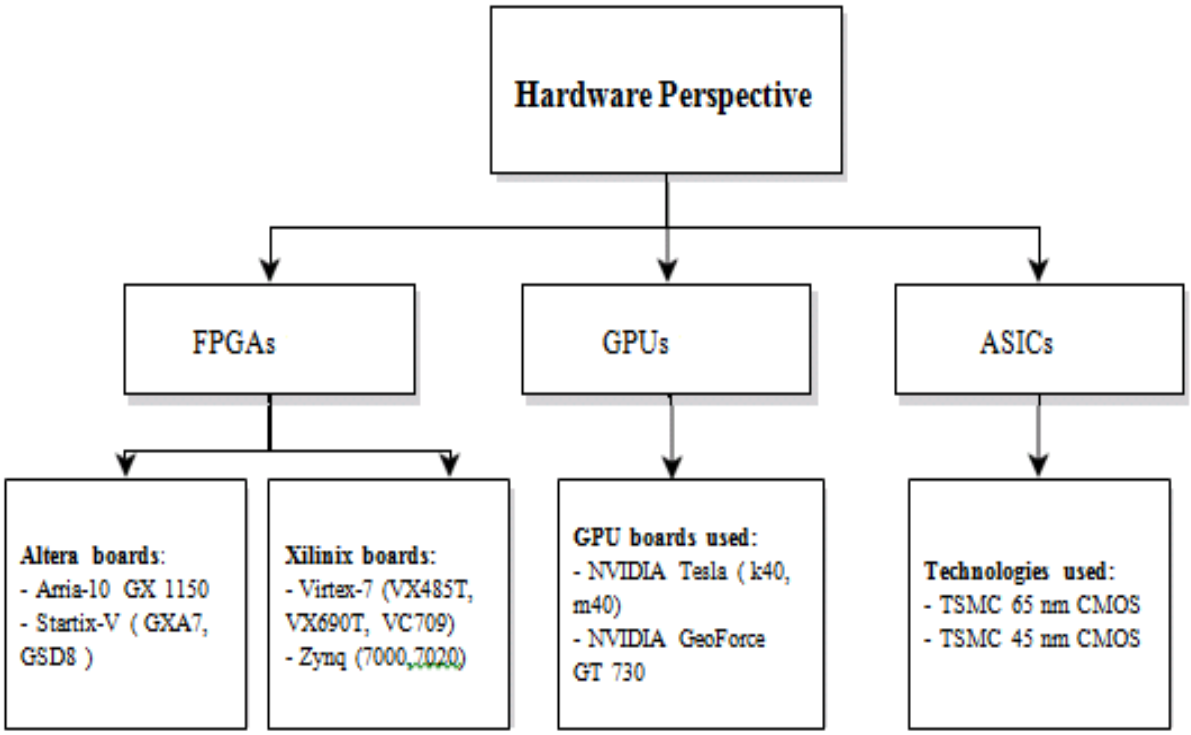


Рисунок 3.3 - Апаратні прискорювачі штучного інтелекту

Розглянемо більш детально сферу застосування кожного з них. Проблемою, що виникає перед дослідниками, є пошук відповідного алгоритму для конкретного додатка та його ефективне нанесення на апаратне забезпечення. CNN будуються з використанням мереж розповсюдження з прямим передаванням, які можна вважати великою кількістю подібних скалярних операцій, що передаються на різних етапах. Цей тип обчислень значно пришвидшується при роботі на ПЛІС. Завдання картографування FPGA включають пошук ефективного узгодження між обчислювальною моделлю CNN та моделлю виконання, підтримуваною FPGA. Кілька дослідницьких робіт розглядали ці питання картографування CNN на ПЛІС.

Евристичний алгоритм пошуку та архітектура набору команд потоку даних використовували DNNWEAVER для створення високопродуктивних прискорювачів, що працюють в рамках обмеженого бюджету енергії та вбудованої пам'яті FPGA. DNNWEAVER – це структура, яка автоматично генерує синтезуваний прискорювач для даної пари (DNN, FPGA) за допомогою оптимізованих вручну шаблонів. Структура DeepBurning побудована для спрощення операції відображення різноманітних нейронних мереж у ПЛІС або ASIC. Бібліотечний компілятор RTL для автоматичного створення індивідуальних прискорювачів на основі FPGA для даного алгоритму ШІ.

Можливість прямого апаратного картографування ШІ на ПЛІС розглядається різними дослідниками. Запропонована модель стверджує, що відкриває нові можливості для подальшої оптимізації і може бути розширена на технології ASIC, а також на двійкові нейронні мережі. Існує методологія проектування на основі затримок для відображення ConvNets у ПЛІС. Засіб автоматизації на основі пітона для створення прискорювачів ШІ на основі FPGA є доволі популярним у використанні. Інструмент постійно забезпечує високу пропускну здатність для різних моделей ШІ. Бібліотечний компілятор RTL досяг найкращої продуктивності: 732,36 GOPS, 1604,57 GOPS, 651,49 GOPS і 789,44 для NiN, VGG-16, ResNet-50 та ResNet-152 відповідно на Startix 10 GX2800 FPGA.

Для AlexNet інструмент автоматизації на основі пітона показав найкращу продуктивність, яка становить 274,5 GOPS.

Було запропоновано багато конструкцій архітектури прискорювачів на основі FPGA. Більшість дослідницьких робіт представляли прискорювачі CNN на основі FPGA. З апаратної точки зору ці документи поділяються на дві категорії:

- різнорідна архітектура;
- ПЛІС.

Гетерогенні прискорювачі архітектури використовують високоякісний процесор з FPGA для реалізації високопродуктивного бінаризованого нейронного прискорювача (BNN). Запропонований прискорювач призначений для використання переваг системи Xeon CPU + FPGA. Архітектура FPGA призначена для найбільш обчислювальних частин BNN, тоді як інші частини топології можуть оброблятися центральним процесором. Є ще інші прискорювачі ШІ на базі CPU-FPGA. Інтенсивні обчислювальні та універсальні операції (наприклад, 2D-згортки) були реалізовані в FPGA.

Інші гетерогенні архітектури, що поєднують ASIC з FPGA, зосереджувались на області глибокого навчання з ефективною тензорною матрицею / векторними операціями. Перспективним є застосування архітектури прискорювача з використанням двох ПЛІС. Реалізація прискорювача на базі ШІ FPGA досягла точності 82,8%.

Прискорювачі ШІ на основі Xilinix FPGA мають найвищу продуктивність серед інших реалізацій. Їх реалізація забезпечує найсучасніші результати. Це пов’язано із використанням міжпластового планування, яке дозволило зменшити передачу даних з 6,6 МБ до 0,2 МБ із зменшенням на 97%.

Прискорювачі ШІ на базі FPGA показують, найвищі показники. Ця архітектура досягає високих показників завдяки декільком джерелам паралелізму, інтегрованим у реалізацію ШІ, метою яких є мета досягнення якнайкращого прискорення. Ці методи паралелізму:

1. Паралелізм між шарами: різні шари, які не мають залежності даних, не можуть виконуватися паралельно.

2. Міжвихідний паралелізм: різні карти вихідних характеристик абсолютно незалежні одна від одної. Тому всі їх можна обчислити паралельно.

3. Міжядерний паралелізм: оскільки згортки незалежні для різних вихідних пікселів, можна використовувати паралелізм.

4. Внутрішньоядерний паралелізм: згортка - це набір множень і додавань.

Оскільки кожне множення між вагою в ядрі та пікселем на вхідній карті особливостей не залежить від іншого множення, їх можна виконувати одночасно.

Інші прискорювачі ШІ на базі FPGA були реалізовані з використанням плат altera. Що стосується реалізації з фіксованою точкою, відомо про найвищу ефективність серед інших реалізацій. Вони звернулись до обмеження продуктивності, яке спричинене малою пропускною здатністю вбудованої пам'яті завдяки двовимірному взаємозв'язку між процесорними елементами (ПЕ) та локальною пам'яттю. Це ефективно збільшує ефективну пропускну здатність вбудованої пам'яті. Тоді як для 32 реалізацій з плаваючою комою на платах altera повідомляється про найвищу продуктивність. Методами, які використовувались для досягнення цієї високої продуктивності, було використання вбудованого буфера потоку, який ефективно зберігає вхідні та вихідні карти функцій. Крім того, використовувався підхід векторизації, який досягає понад 60% ефективності DSP.

Також використовуються вбудовані FPGA процесори, такі як NIOS-II, та ARM-процесори в FPGA на основі SoC для реалізації цільових прискорювачів. Також прискорення ШІ може досягатися за допомогою використання розпаралельованої програмованої логіки для реалізації згортки, об'єднання та доповнення, а також вбудованого процесора ARM для роботи та виконання решти завдань.

Графічні процесори стають важливими для прискорення глибокого навчання. Існуючі методи спрямовані на прискорення алгоритмів на графічних процесорах.

Протягом останнього десятиліття співпроцесор Intel Xeon Phi проклав шлях до успіху у впровадженні алгоритмів глибокого навчання. Intel Xeon Phi - це спільна пам’ять. Це означає багатоядерний співпроцесор, який містить до 61, 1,2 ГГц ядра, і кожне ядро може перемикатися між 4 апаратними потоками круговим способом. Xeon Phi широко використовує технологію Single Instruction Multiple Data (SIMD), яка дозволяє виконувати одну і ту ж операцію на кількох сегментах даних одночасно. Таким чином, Xeon Phi дуже підходить для програм глибокого навчання, які зазвичай використовують прості операції над великими тензорами.

Дослідники довели, що Intel Xeon Phi може запропонувати ефективний, але більш загальний спосіб паралелізації алгоритму глибокого навчання порівняно з графічними процесорами. Глибоке навчання можна ефективно оптимізувати та масштабувати на багатоядерних системах HPC.

Запропонований CosmowFlow, який є першим широкомасштабним науковим застосуванням рамки TensorFlow у суперкомп'ютерному масштабі з повністю синхронним. Оптимізований повний стек програмного забезпечення, що включає дизайн мережі, конвеєр обробки вводу-виводу, зв'язок, фреймворк TensorFlow та примітиви 3D CNN в MKLDNN для 3D-згорткової нейронної мережі. Деякі дослідники зосереджувались на BNN, де був запропонований новий підхід для оптимізації BNN на процесорах за допомогою фреймворку BitFlow. BitFlow отримує 1,8 прискорення.

Апаратні технології для машинного навчання:

- програмований FP-DNN, структура, що має вхідні дані описуваних TensorFlow DNN, що використовуються для генерації апаратних реалізацій на платах FPGA з гібридними шаблонами RTL-HLS;

- структура, що використовується для автоматизації відображення ШІ на систолічних масивах на ПЛІС;

- TuRF структура прискорення ШІ, натхненна ефективними архітектурами ШІ та навчанням трансферу, що підтримує оптимізацію для конкретного домену. Він ефективно використовує доменні програми на FPGA;

- Caffe використання відкритого коду системи глибокого навчання для забезпечення чіткого доступу до глибоких архітектур. Код написаний на чистому, ефективному C ++, де CUDA використовується для обчислення графічного процесора;

- clCaffe, прискорення OpenCL добре відомого механізму глибокого навчання Caffe, при цьому фокусуючись на рівні згортки, який був оптимізований трьома різними підходами. Це значно покращує можливість використання випадків глибокого навчання на всіх типах пристроїв OpenCL.

Існує багато інструментів з відкритим кодом, що підтримуються NVIDIA та Intel для підтримки прискорення алгоритмів машинного навчання. Наприклад, NIVIDIA розробила RAPIDS, це бібліотека машинного навчання з прискореним графічним процесором, яка має на меті значно пришвидшити навчальний процес. RAPIDS покращує продуктивність, прискорюючи повний конвеєр робочого циклу, включаючи підготовку даних, алгоритм машинного навчання та візуалізацію на графічних процесорах. RAPIDS також дозволяє значно прискорити обробку, навчати більшим розмірам наборів даних та підтримує розгортання декількох графічних процесорів. Швидкий використований для успішного скорочення необхідного часу навчання систем, що рекомендують, що обчислювально вважається інтенсивним із прискоренням у 15,6 рази.

Проведений аналіз показав, що для конкретної задачі, що розглядається в даній роботі, доцільним для прискорення навчання ШІ є використання моделі прискорювача штучного інтелекту, що використовує процесор Intel I9 11900 (11 покоління), який мав нову архітектуру для штучного інтелекту – Intel Deep Learning Boost.

3.2 Алгоритми функціонування інтелектуалізованої системи

Архітектура підсистеми взаємодії агента з зовнішнім середовищем наведена на рисунку 3.4.

Зовнішнє середовище впливає на зміну станів агента. В свою чергу формуються змінні станів, що взаємодіють з інтерфейсом агента. Інтерфейс взаємодіє з базою навичок агента, яка, в свою чергу, є частиною так званої бази знань агентного середовища. Результати взаємодії та опрацювання всієї наявної інформації передається на інтерпретатор знань та навичок агента. Результати інтерпретації в подальшому використовуються для формування множини стратегій для навчання агента.

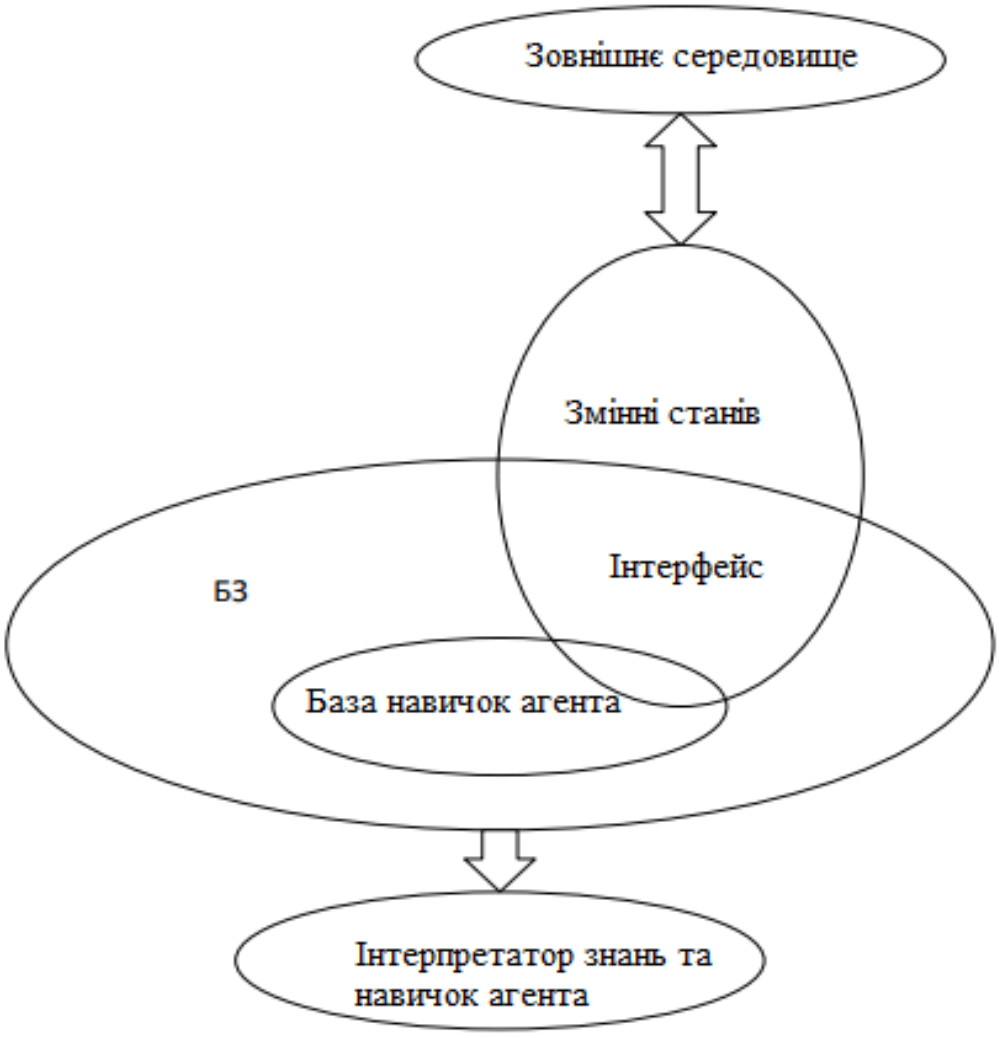


Рисунок 3.4 – Архітектура підсистеми взаємодії агента з зовнішнім середовищем

На основі удосконаленого агентно-орієнтованого методу розроблений алгоритм, що реалізовує даний метод. Алгоритм представлений на рисунку 3.5.

На початку формується початковий набір значень для навчання агента. Після того відбувається перевірка чи агент виконав дію. Якщо дія була виконана, то відбувається процедура отримання вагового коефіцієнта винагороди. Далі включається лічильник підсумовування вагових коефіцієнтів винагороди та занесення даних в таблицю.

Якщо агент не виконав дію або за інших причин дія не відбулася, то повертаємось до повторного кроку навчання агента на вибраному наборі значень.

Рисунок 3.5 – Алгоритм агентно-орієнтованого методу

Функційна схема, що описує процес навчання агента з використанням агентно-орієнтованого методу наведена на рисунку 3.6. Основним моментом тут є можливість створення та запам’ятовування ігрових стратегій агента, використовуючи аналіз ігрових дій агента. Після аналізу дії агента відбувається прийняття рішення щодо реакції агента на основі марківського процесу та формування набору вагових коефіцієнтів винагороди, що дає змогу найбільш

повно врахувати всі дані, які використовують при навчанні з підкріпленням агента. Після цього відбувається перевірка чи агент зреагував на зміну стану, і на основі результатів цієї перевірки відбувається або додавання нової стратегії до множини дій агента, якщо агент зреагував на зміну стану, виконавши дію, або припинення процесу навчання агента.

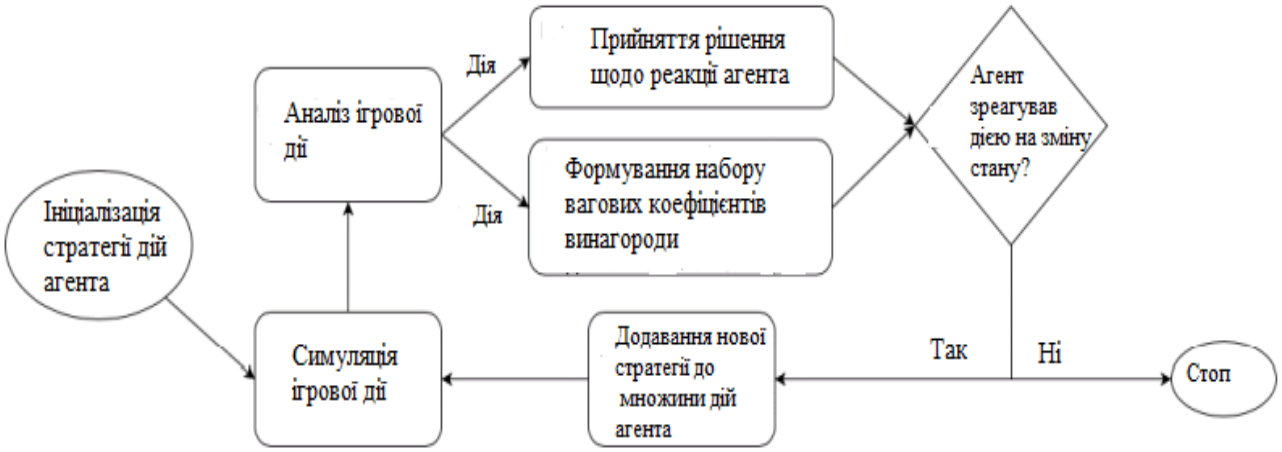


Рисунок 3.6 – Функційна схема, що описує процес навчання агента з використанням агентно-орієнтованого методу

Функція початку дії агента представлена на рисунку 3.7, для її візуалізації використано безпосередньо середовище для створення комп’ютерних ігор.

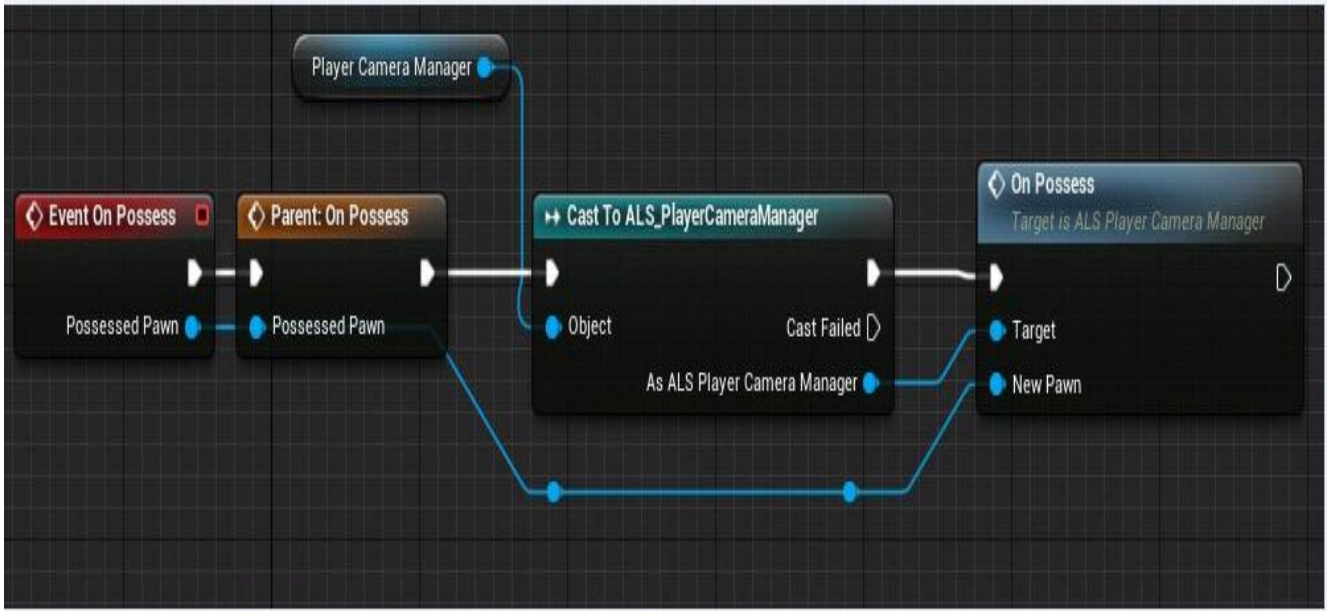


Рисунок 3.7 – Функція початку дії агента

Функція обрахунку результату дії одного агента над іншим, є лише 2 результати:

1). нанесено удар;

2). заблоковано удар.

Дерево поведінки агента на карті представлено на рисунку 3.8.

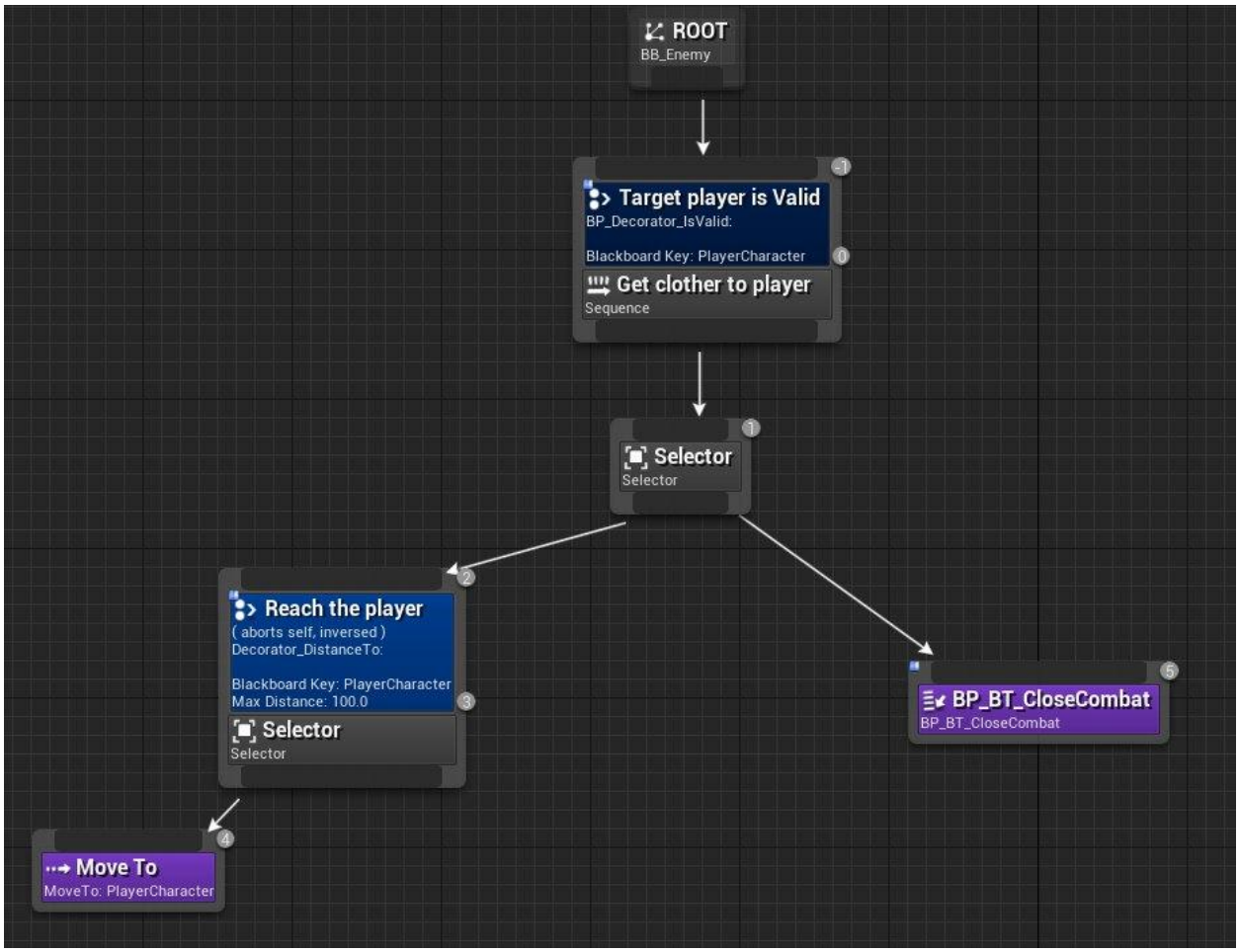


Рисунок 3.8 – Дерево поведінки агента на карті

Дерево поведінки представляє собою орієнтований ациклічний граф, який має можливі варіанти введення роботи. «Ширина» дерева вказує на кількість доступних дій, а «довжина» його гілок характеризує їх складність. В даному випадку дерево поведінки персонажу не є складним, тобто ні в ширину, ні в довжину не є великим за розміром. Це пов’язано зтим, що стандартна поведінка персонажу є доволі простою. Він взаємодіє з зовнішнім середовищем, змінює свої стани, очікує впливу зовнішнього середовища або оякує дії чи протидії, або їх виконує в залежності від конкретної ситуації.

Результат роботи будь-якого вузла завжди передається батьківського вузла, розташованого на рівень вище. Дерево проглядається з самого верхнього вузла до кореня. Від нього виробляється пошук в глибину починаючи з лівої гілки дерева. Якщо у одного вузла є кілька підзадач, вони виконуються зліва направо.

Серед вузлів виділяють наступні типи:

- дія (action),
- вузол виконання послідовності (sequence),
- паралельний вузол (parallel),
- селектор (selector),
- умова (condition),
- інвертор (inverter).

Дія являє собою запис змінних або який-небудь рух. Вузли послідовностей по черзі виконують поведінки кожного дочірнього вузла доти, поки один з них не видасть значення «Невдача», «В роботі» або «Помилка». Якщо цього не відбулося, повертає значення «Успіх».

Вузли паралельних дій виконують поведінки дочірніх вузлів до тих пір, поки задану кількість з них не поверне статуси «Невдача» або «Успіх».

Селектори по черзі виконують поведінки кожного дочірнього вузла доти, поки один з них не видасть значення «Успіх», «В роботі» або «Помилка». Якщо цього не відбулося, повертає значення «Невдача».

Стандартне дерево поведінки для NPC представлено на рисунку 3.9.

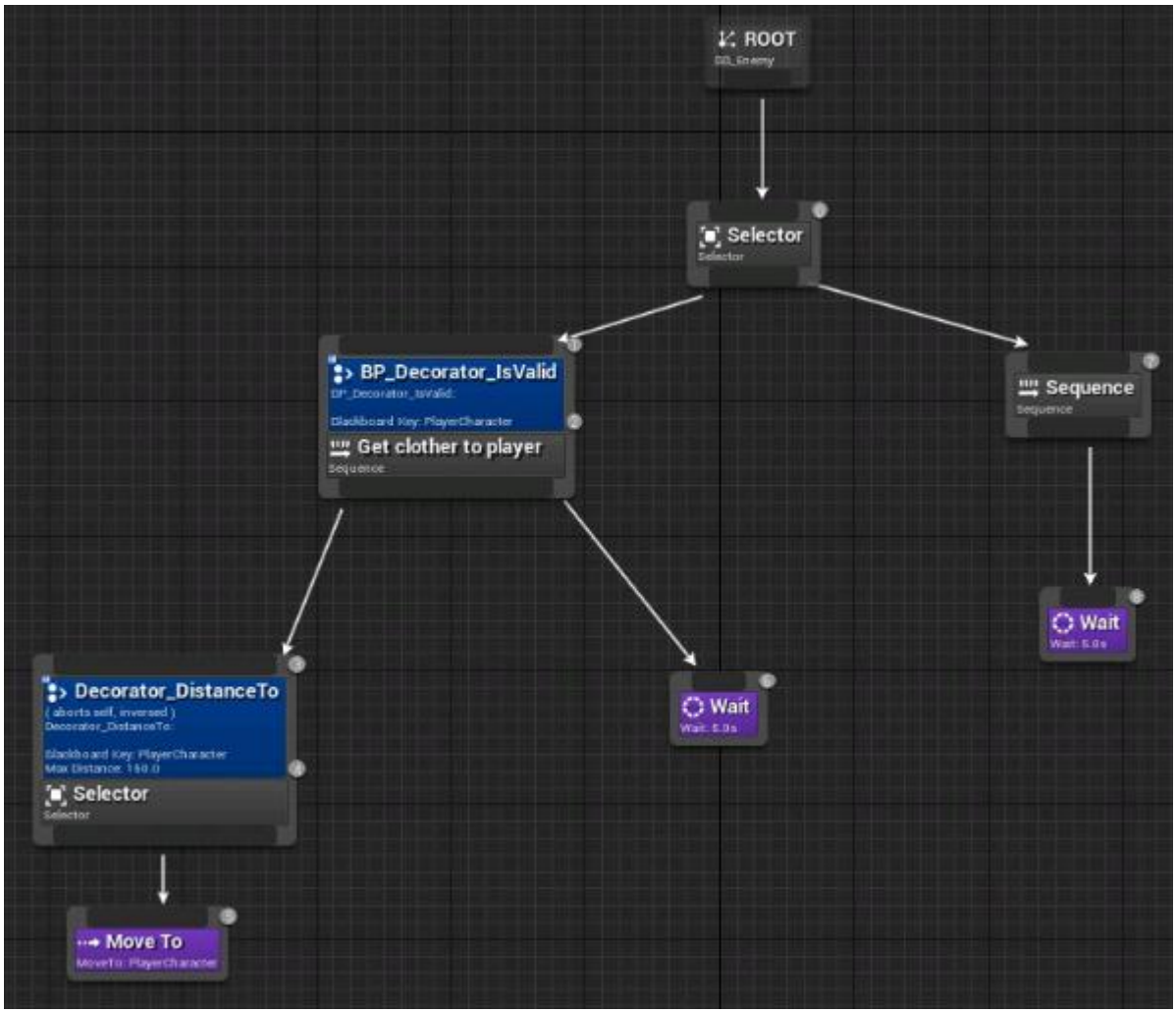


Рисунок 3.9 – Стандартне дерево поведінки для NPC

Спочатку агент шукає гравця, після чого підходить до нього, як тільки ціль досягнута – починається бій і вмикається цикл з QTable.

Алгоритм нанесення персонажем удару представлений на рисунку 3.10. Після того як гравець натиснув удар,виконується запит для відтворення дії для виконання удару.

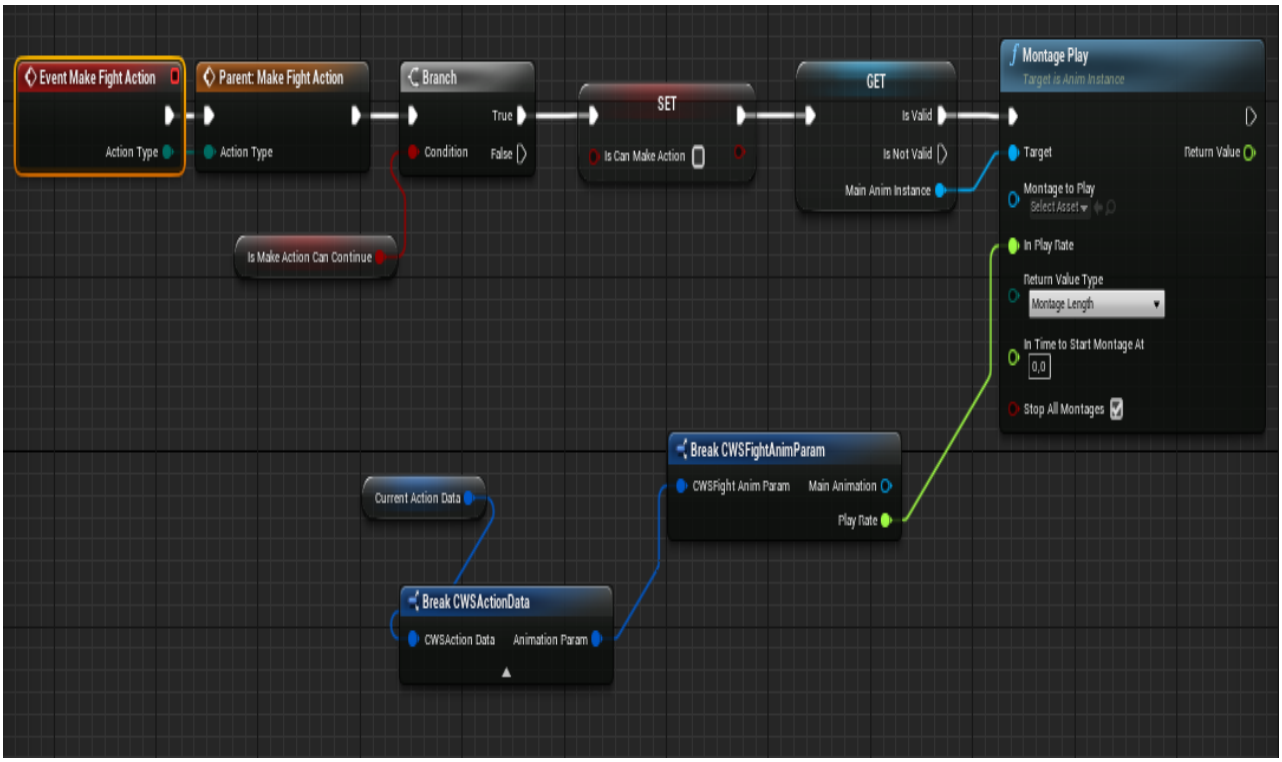


Рисунок 3.10 – Алгоритм нанесення персонажем удару

Якщо проблемний домен особливо складний, великий або непередбачуваний, тоді єдиний спосіб, з яким його можна розумно вирішити, - це розробка ряду функціонально конкретних та модульні компоненти (агенти), які спеціалізуються на вирішенні певного аспекту проблеми. Це розкладання дозволяє кожному агенту використовувати найбільш підходящу парадигму для вирішення її конкретної проблеми. Коли виникають взаємозалежні проблеми, агенти в системі повинні координувати між собою, щоб забезпечити належне управління взаємозалежностями Основними підходами до імітаційного моделювання є системна динаміка, дискретна подія та імітаційне моделювання на основі агента. Системна динаміка та дискретні моделювання подій є традиційними; тоді як агентське моделювання - відносно новий підхід. На основі агента моделювання корисно на всіх рівнях абстракції та може моделювати великі складні системи.

Функційна схема роботи штучного інтелекту під час бою наведена на рисунку 3.11.

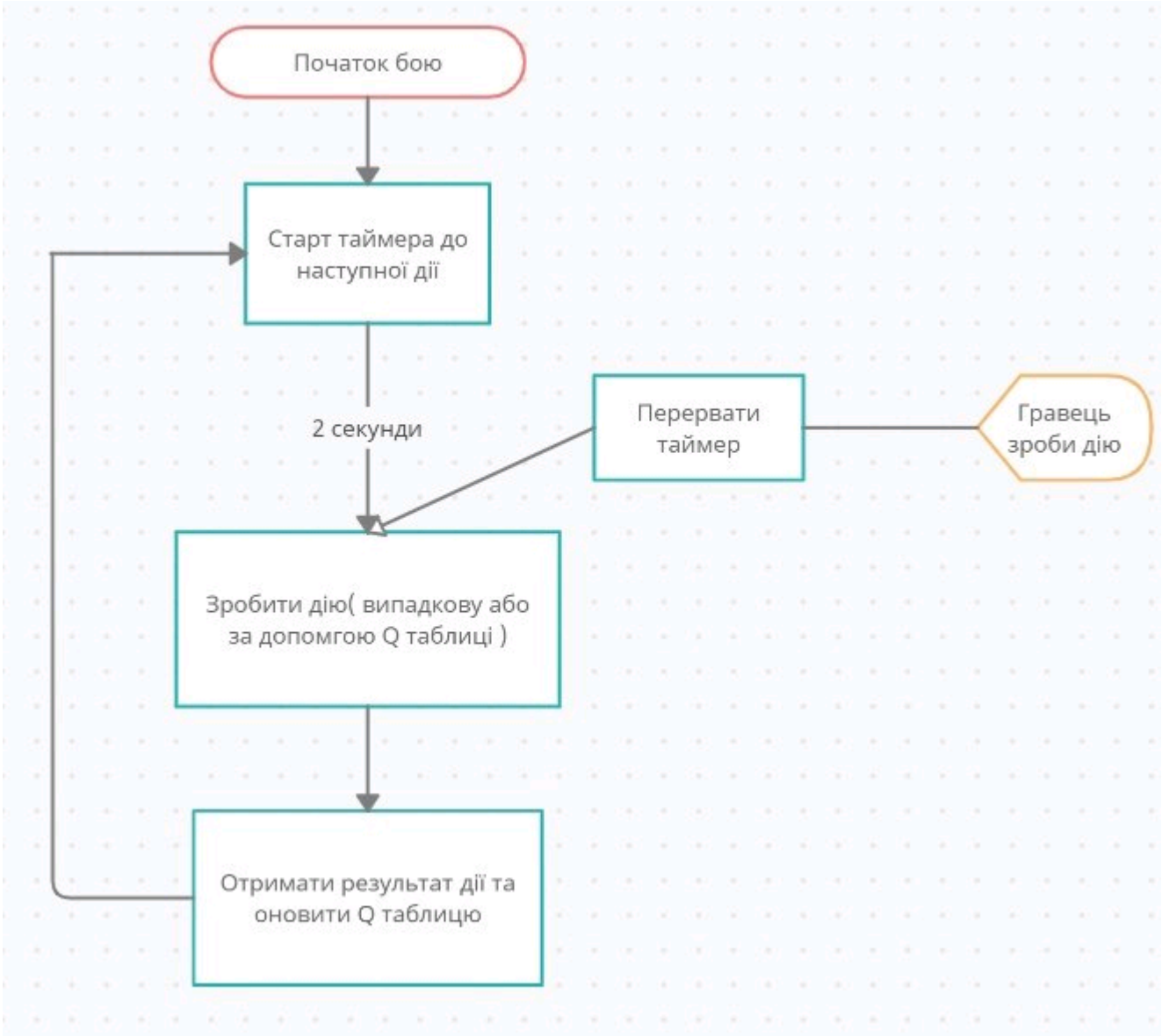


Рисунок 3.11 – Функційна схема роботи штучного інтелекту під час бою

Алгоритм дій роботи штучного інтелекту під час бою полягає в тому, що спочатку штучний інтелект шукає гравця, після цього оцінює відстань до нього, далі скорочує дистанцію. Після того як відбулась перевірка чи дистанція скоротилась, активується машинне навчання з підкріпленням і відбувається бій. Після якого результати мають бути збережені для формування бази стратегій і подальшого машинного навчання.

Алгоритм протидії агента представлений на рисунку 3.12.

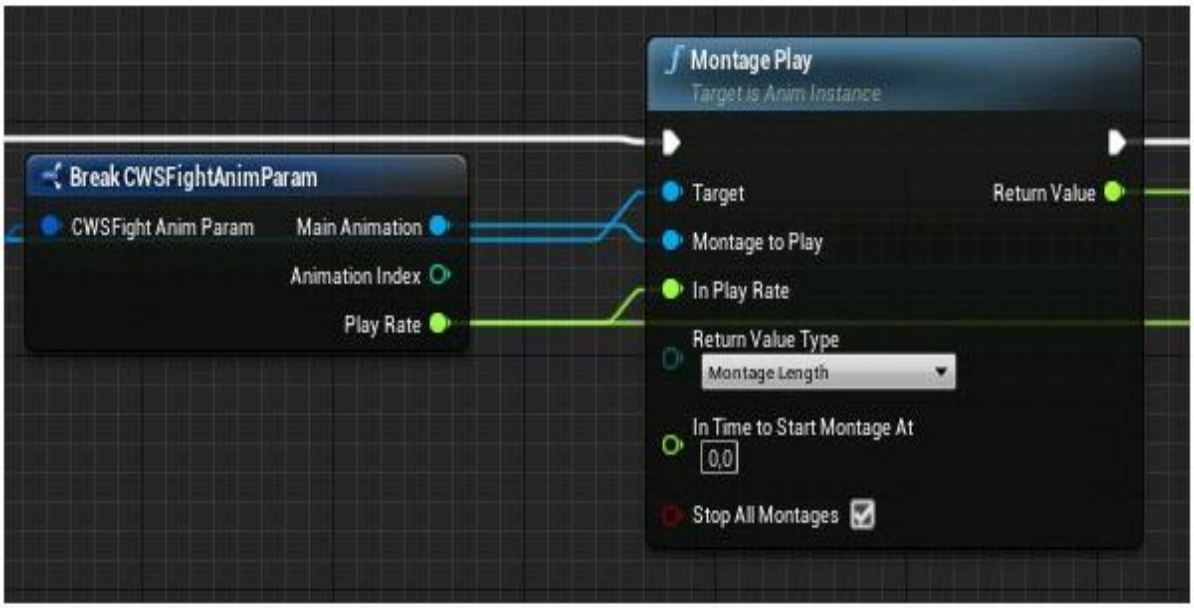


Рисунок 3.12 – Алгоритм протидії агента

Для кожної дії є протидія, наприклад, якщо гравець вдарив лівою рукою, то ШІ мусить поставити блок правою, тоді він не отримає удар.

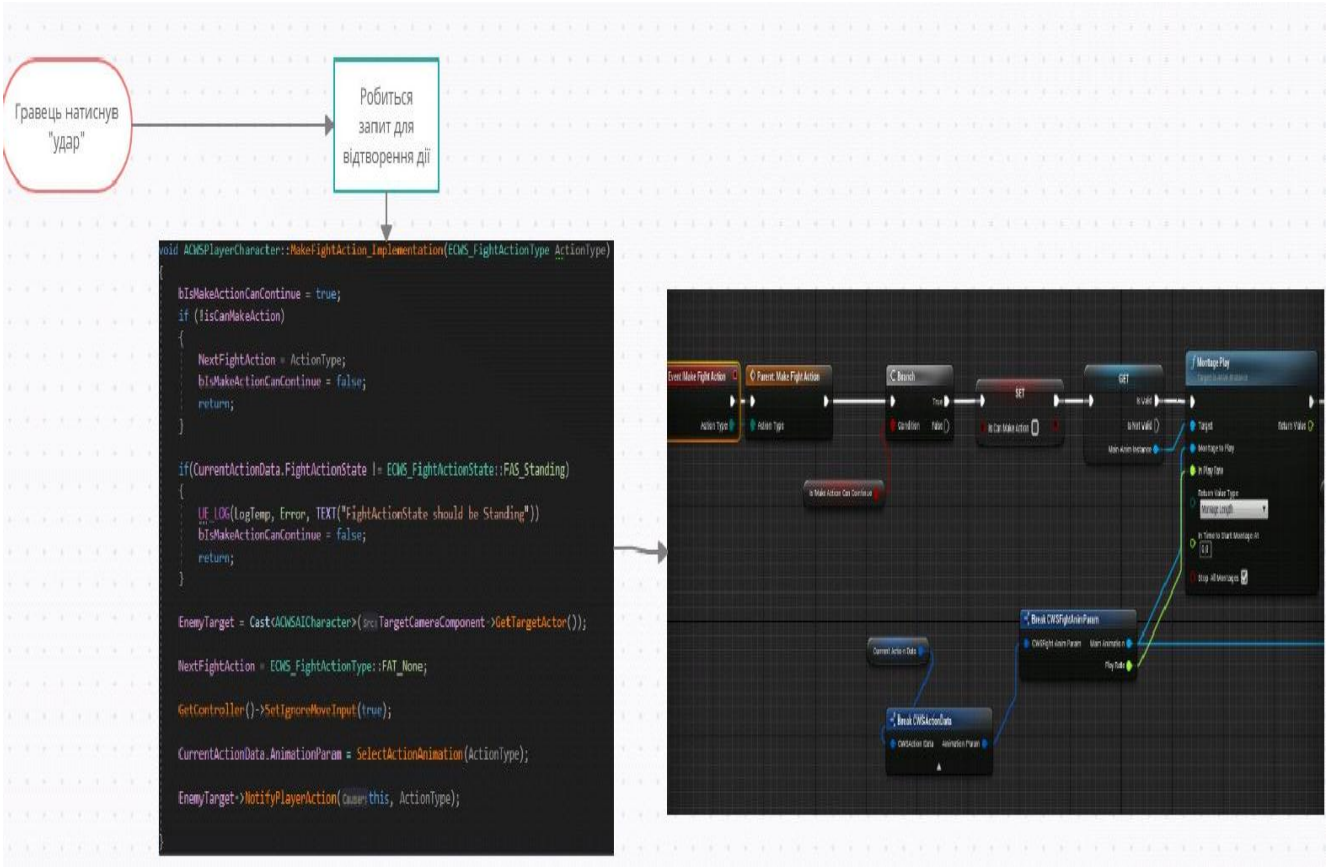


Рисунок 3.13 – Алгоритм дії агента під час виконання удару

В загальному алгоритм дії агента виглядає наступним чином:

1). функція початку дії агента у коді;

2). розширена функція дії у блупринтах;

3). початок циклу QTable з інтервалом в 2 секунди;

4). якщо гравець здійснює дію – таймер переривається и одразу іде обрахунок контр-дії;

5). штучний інтелект обирає яку зробити дію(випадкову або обдуманну) за допомогою випадкового числа;

6). обирається найоптимальніша дія;

7). бали за результат дії;

8). оновлення таблиці після результату дії.

Фрагмент коду, який відповідає за нарахування балів за результат дії наведений на рисунку 3.14.

```
void UCWSMachineLearningComponent::UpdateQTable(FCWSFightActionResultData ResultData)
{
	OldAction = CurrentAction;
	OldState = CurrentState;
	CurrentState = 0;
	float DeltaError = GetReward(ResultData) + gamma * GetMaxQAction(CurrentState) -
QTable[OldState][OldAction];
	QTable[OldState][OldAction] = QTable[OldState][OldAction] + (alpha* DeltaError);
	SaveToDataTable();
}
```

Рисунок 3.14 - Фрагмент коду, який відповідає за нарахування балів за результат дії

3.3 Висновки

У третьому розділі роботи, відповідно до розробленої формалізованої моделі агента та агентно-орієнтованого методу, виконано проектування структури ПЗ та розроблено алгоритм роботи інтелектуалізованої системи на основі методів машинного навчання для розроблення комп’ютерних ігор. Проведений аналіз показав, що для конкретної задачі, що розглядається в даній роботі, доцільним для прискорення навчання ШІ є використання моделі прискорювача штучного інтелекту, що використовує процесор Intel I9 11900 (11 покоління), який мав нову архітектуру для штучного інтелекту – Intel Deep Learning Boost. У процесі проектування структури ПЗ виконано необхідні уточнення структури підсистеми машинного навчання, яка описана у формалізованій моделі.

4 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вибір засобів розроблення ПЗ інтелектуалізованої системи

Для програмної реалізації комп’ютерної гри був використаний Unreal Engine 4. Це набір інструментів для розробки ігор, який має широкі можливості: від створення двохмірних ігор на мобільних пристроях до AAA-проєктів на консолі. Цей двигун використовувався при розробці таких ігор як ARK: Survival Evolved, Tekken 7, Kingdom Hearts III. Для початківців розробка в Unreal Engine 4 є дуже простою. За допомогою системи візуального створення скриптів Blueprints Visual Scripting можна створювати готові ігри, не пишучи зовсім код. Перевагами є простота використання, зручний інтерфейс, безкоштовність.

Перша гра, яка була створена на цьому двигуні з’явилась в 1998 році і називалась Unreal. З тих пір і до сьогоднішнього часу різні версії двигуна були використані більше ніж в сотні ігор та інших проєктів.

Unreal Engine 4 – ігровий двигун, що надає широкі можливості для програмістів, гейм дизайнерів та художників. 4-а версія двигуна з’явилася відносно недавно, на цій версії був сворений відомий проєкт Dead Island 2. Третя версія цього двигуна була доволі популярною і на ній створений цілий ряд відомих комп’ютерних ігор. Наприклад, BioShock Infinate, XCOM: Enemy Unknown, Batman: Arkham City, Bulletstorm, Mortal Kombat, трилогія Mass Effect та багато інших відомих ігор.

Цей двигун написаний на мові C++, він дозволяє створювати ігри для більшості операційних систем та платформ: Microsoft Windows, Linux, Mac OS, Mac OS X; консолей Xbox, Xbox 360, PlayStation 2, PlayStation 3, PSP, PS Vita, Wii, Dreamcast, GameCube та ін., а також на різних портативних пристроях. Наприклад, пристроях Apple (iPad, iPhone), що керуються системою iOS та інше. Вперше робота з iOS була представлена в 2009 році, а в 2010 році була продемонстрована робота двигуна на пристрої з системою webOS.

Для того, щоб спростити портотування, двигун використовує модульну систему залежних компонентів, підтримує різноманітні системи рендерінгу

(Direct3D, OpenGL, Pixomatic, в більш ранніх версіях: Glide, S3, PowerVR), відтворення звуку (EAX, OpenAL, DirectSound3D, раніше A3D), засоби голосового відтворення тексту, розпізнавання мови, модулі для роботи з мережею та підтримки різноманітних пристроїв введення.

Для ігор по мережі підтримуються технології Windows Live, Xbox Live, GameSpy та інші, є можливість приєднувати до 64 гравців одночасно. Таким чином, двигун адаптували для застосування в іграх жанру MMORPG (один з прикладів Lineage II).

В цьому двигуні можна писати ігрову логіку на C++, а також з допомогою візуальної системи програмування Blueprint.

Рекомендовані системні вимоги для використання ігрового редактора:

- Desktop PC або Mac;
- Windows 10 64-bit або Mac OS X 10.9.2 або вище;
- Quad-core Intel або AMD процесор, 2.5 GHz або швидше;
- 8 GB RAM.

Але насправді мінімальні вимоги нижче, завдяки гнучкому настроюванню графіки.

Самий ігровий редактор складається з декількох функціональних частин: гра, двигун, плагіни та платформи. В центральній частині інтерфейсу знаходиться основне вікно Viewport, в якому будуть розміщуватись всі ігрові об’єкти. Можна відкривати одразу декілька Viewport’ів, але це буде потребувати додаткових системних потужностей. Над цим вікном знаходиться панель Toolbar, за допомогої котрої здійснюється керування проєктом. В лівій частині знизу розташоване дерево контенту Content Browser (звуки, моделі, текстури, анімація та інше), а зверху вікно Modes, за допомогою якого можливо додавати різноманітні геометричні фігури, джерела освітлення, елементи ландшафту та інші речі. В правій верхній частині відображене вікно SceneOutliner, яке враховує всі об’єкти на сцені. В правому нижньому куті знаходиться вкладка Details, де можна подивитись та виправити властивості окремих об’єктів.

Вся ігрова логіка пишеться на візуальній скриптовій мові Blueprint. Також можна підключати чисту мову C++. Зв'язок об’єктів, генерація рівнів, система реманентів, здоров’я та ушкодження персонажів у грі – все це прописується у формулах та коді.

4.2 Розроблення ПЗ інтелектуалізованої системи

Інтелектуалізована система складається з наступних модулів. Структурна схема розробленої системи представлена на рисунку 4.1.

MyGame – Основний модуль в якому проводилися тести.

Engine – Основний модуль ігрового двигуна Unreal Engine.

Editor – Модуль відповідає за редактор ігрового двигуна.

Core – Головний модуль, який обробляє всю ігрову логіку

InputCore – Модуль обробки вхідних даних, таких як – натискання клавіш гравцем.

CoreUObject – Модуль для роботи з рефлексією Unreal Engine 4.

Fight System – Модуль відповідає за бойову систему гри, тут реалізовані всі дії для гравця і агента.

AIModule – Головний модуль штучного інтелекту, відповідає за навігацію і стан дій комп’ютера.

Machine Learning – Модуль відповідає за машинне навчання з підкріпленням, а саме QTable.

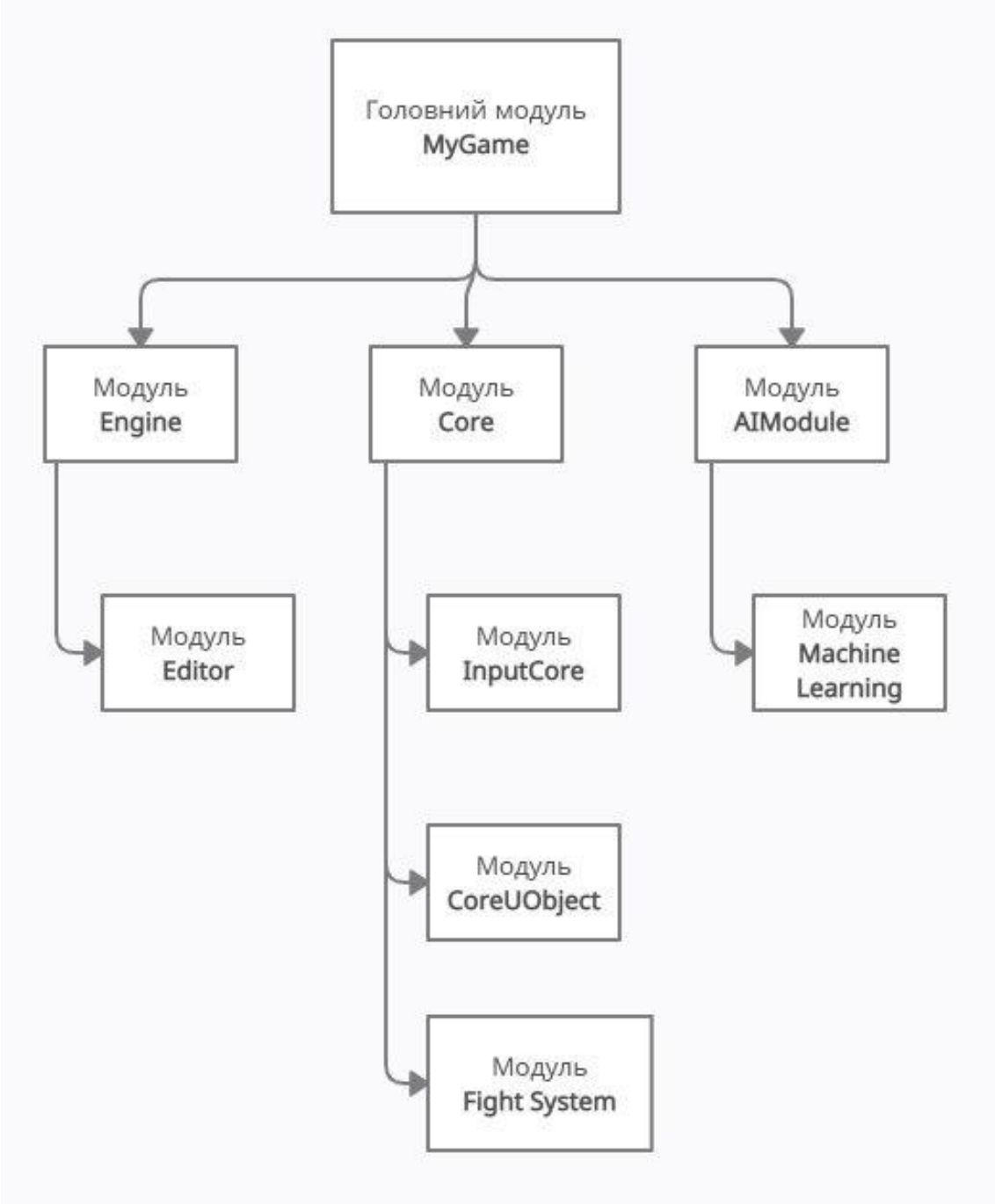


Рисунок 4.1 – Структурна схема інтелектуалізованої системи

Розглянемо більш детальніше основну функційність модулів системи.

MyGame – Під час запуску перевіряє наявність і коректніть роботи модулів.

AIModule – Даний модуль містить усю логіку для взаємодії штучного інтелекту з оточенням. Він розміщує NavMesh, по якому агент орієнтується і може пересуватись, також обирає дію згідно дерева поведінки, яке містить лише поточні стани із яких агент вибирає дію.

Спочатку, штучний інтелект шукає гравця за допомогою отримання поточного контролера(GetController()), після чого будує маршрут і підходить до нього, як тільки умова виконана – запускається модуль Machine Learning.

Machine Learning – Модуль під час старту оновлює данні із таблиці у форматі CSV, або ініціалізує нову з 0(залежить від вибору у настройках), після чого очікує команду до старту, після чого запускає таймер, який кожні 2 хвилини робить дію(випадкову, або обдуманнну згідно QTable), також, якщо гравець робить дії по відношенню к штучному інтелекту, таймер привається і агент обирає контр-дію, після чого оновлює таблицю і знову запускає таймер.

Engine – Робить за нас увесь графічний пайплайн (від точок до повністю готової картинки), об'єднує усі необхідні модулі для розробки і передає необхідний функціонал редактору.

Editor – Має логіку коммунікації між користувачем і двигуном, виводить усю інформацію за допомогою Slate і має внутрішню ієрархію. Також приймає усі виклики вводу від користувача.

Core – Об’єднує усі модулі і проводить логіку під час, перевіряє, яка саме платформа використовується і зазделегіть підставляє необхідний код за допомогою макросу #ifdef.

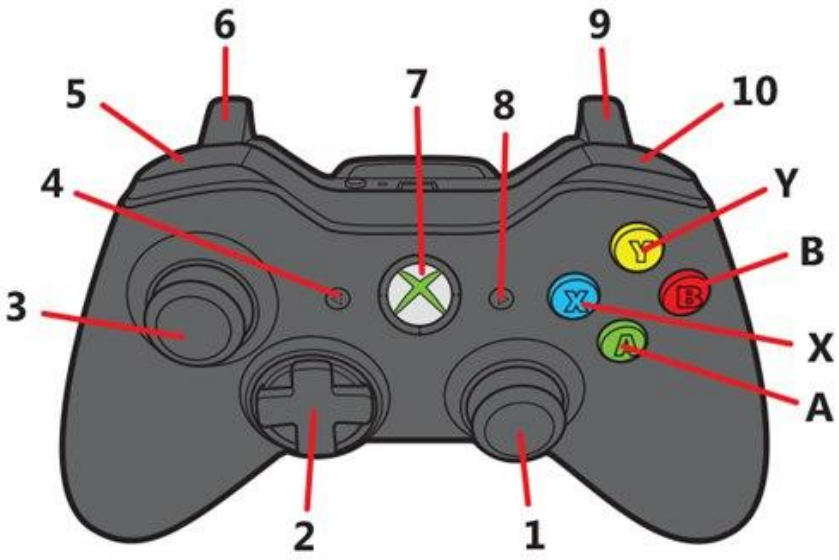


Рисунок 4.2 – Пульт керування геймпадом

Керування розраховано для геймпадів, наприклад візьмемо контролер від ігрової консолі XBOX 360 минулого покоління.

Для пересування персонажу використовується лівий аналоговий стік (3), а для керування камерою – правий аналоговий стік (1), також, якщо натиснути на цей стік, камера перейде у режим бою і візьме на ближайшого ворога(агента) по центру.

Лівий бампер(5) використовується для блокування лівою рукою рукою, якщо зажати і зняти блок, якщо кнопку відпустити, відповідно до правого бамперу (10), який має аналогічний функціонал, але для правої руки.

Лівий і правий трігер(6 і 9) містять у собі 4 команди.

Якщо повністю зажати тригер – гравець зробить атаку лівою чи правою рукою відповідно до обраного трігеру, якщо ж трігер нажати лише наполовину(~45%), то гравець зробить обманний замах рукою, що винудить опонента подумати, що зараз буде зроблено удар.

Кнопка START (8) використана для виклику меню під час гри.

Важливою складовою інтелектуалізованої системи на основі методів машинного навчання для розроблення комп’ютерних ігор є підсистема машинного навчання. Простір дій у грі графа-атаки величезний для нападника та захисника, оскільки захисник може захищати будь-яку підмножину вузлів і таким чином агент змінює свій стан. Нападник може обрати будь-яку підмножину вузлів. Для того, щоб зробити експоненціальний простір дій простірним, - це дозволити кожному агенту додавати предмети, на які потрібно здійснити атаку чи захист, по одному до набору атак чи набору захисту. Наприклад захисник спочатку на кожному часовому кроці встановлений захист порожній, а захисник може або додати один вузол до набору, або пройти. Зрештою, коли захисник проходить, гра триває із захисником, що діє на вузли в наборі захисту. Таким чином, агент викликається кілька разів протягом одного часового кроку, щоб визначити набір елементів, на які потрібно діяти.

Також була розроблена підсистема машинного навчання інтелектуалізованої системи. Функційна схема підсистеми машинного навчання інтелектуалізованої системи зображена на рисунку 4.3.

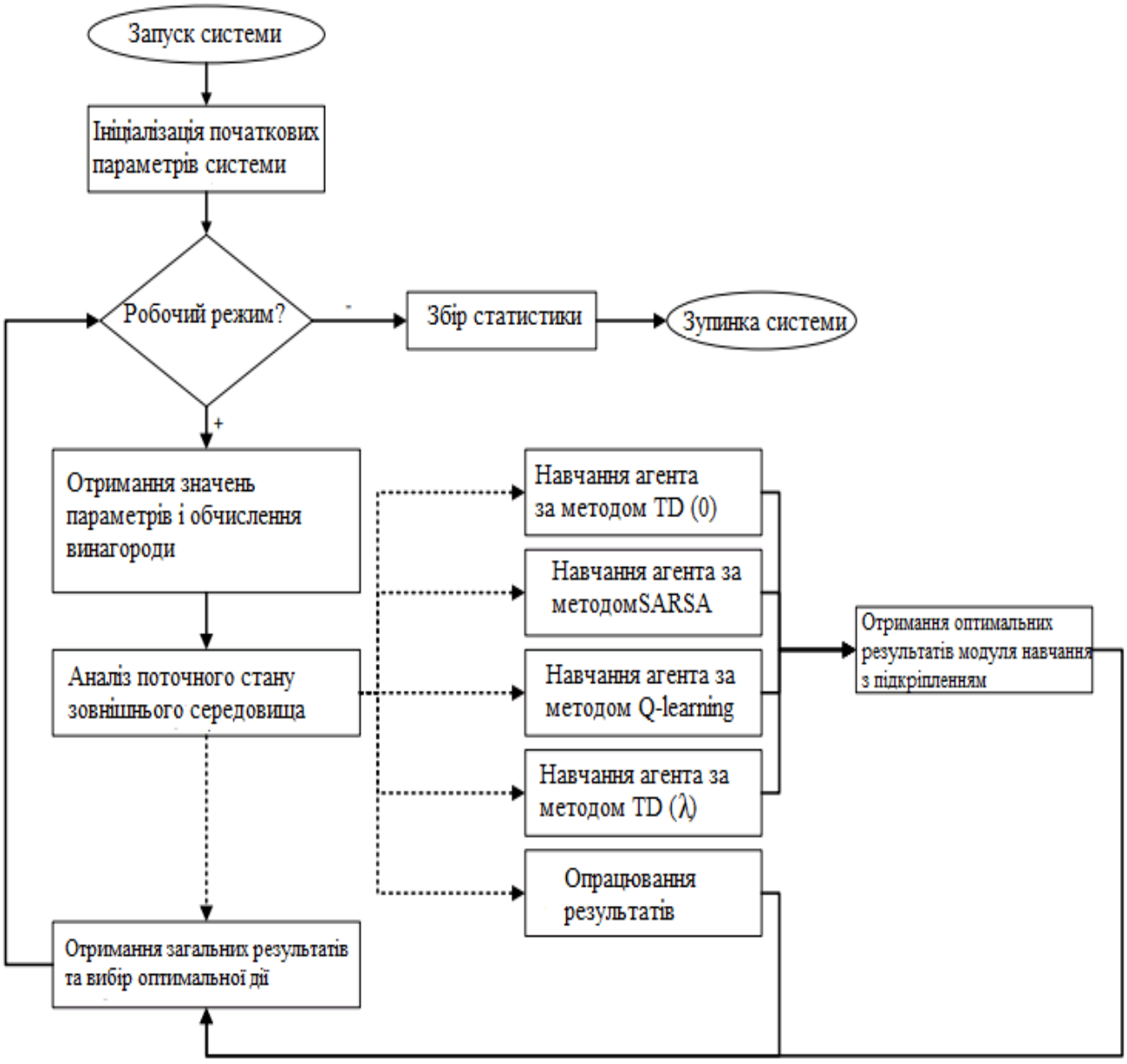


Рисунок 4.3 – Функційна схема підсистеми машинного навчання інтелектуалізованої системи

В даній підсистемі є можливість з розвитком інтелектуалізованої системи інтегрувати методи прогнозування з використанням статистичних та експертних методів, алгоритмів TD-методів навчання та гнучкого алгоритму пошуку рішень,

що здатна отримувати результати в різних умовах, в тому числі в реальному часі та обмеженому часовому просторі. В ідеальному варіанті при наявності достатнього часу та ресурсів виконується паралельне отримання значень параметрів та аналіз поточного стану зовнішнього середовища. В умовах жорстких часових та ресурсних обмежень можна застосувати технологію розбиття на етапи згідно якої алгоритм обирає найбільш перспективний відносно дій агента та обчислення винагороди шлях та розраховує результат найбільш підходящими та адекватними на даний момент методами. При цьому всі інші етапи можуть виконуватись у фоновому режимі з метою включення їх в аналіз та статистику на наступних кроках. В подальшому в підсистемі машинного навчання можливо інтегрувати методи машинного навчання на основі темпоральних різниць від найбільш простого методу TD(0) до більш складних методів, SARSA – з інтегрованою оцінкою цінності стратегії, Q-навчання – з розділеною оцінкою цінності стратегії, TD(λ) – з часовою різницею тривалістю n кроків. На рисунку пунктиром позначені необов'язкові етапи, виконання яких може вар'юватися відносно поточного стану системи. Інтеграція методів дозволяє реалізовувати наступні переваги:

- гнучкість та адаптуємість, в залежності від стану середовища можуть використовуватись різноманітні паралельні алгоритми навчання та прогнозування та їх комбінації;
- можливість розрахунку передбачуваної дії незалежно від доступних часу та ресурсів пам'яті;
- можливість практично одразу видачі повідомлення про рішення за запитом і продовження розрахунків у фоновому режимі;
- знаходження найбільш ефективних алгоритмів для поточного середовища та знаходження найкращих локально або глобально оптимальних рішень.

Актор-агент, No Playable Character з яким навчається штучний інтелект представлений на рисунку 4.4. Приклад анімації персонажу представлений на рисунку 4.5.

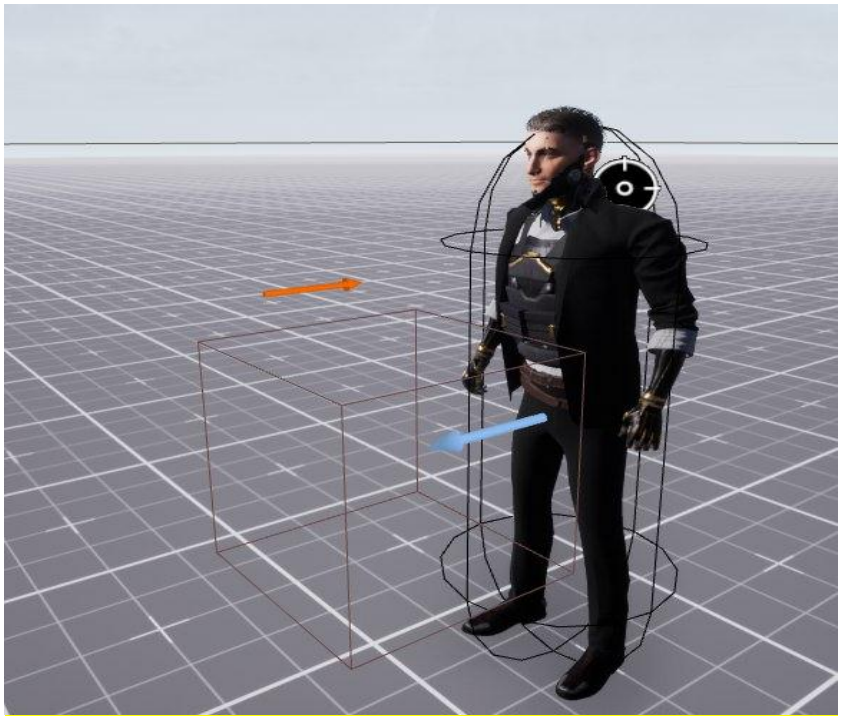


Рисунок 4.4 – Приклад анімації актора-агента

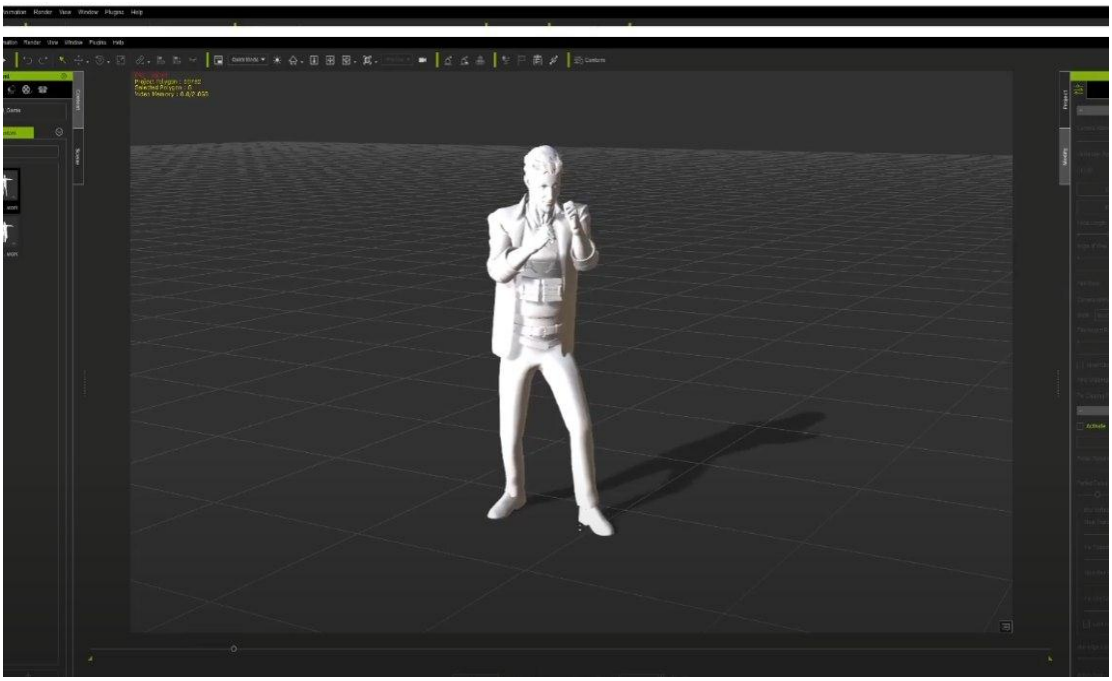


Рисунок 4.5 – Приклад анімації персонажа

Приклад анімації персонажу з тригером оберненої дії представлений на рисунку 4.6.

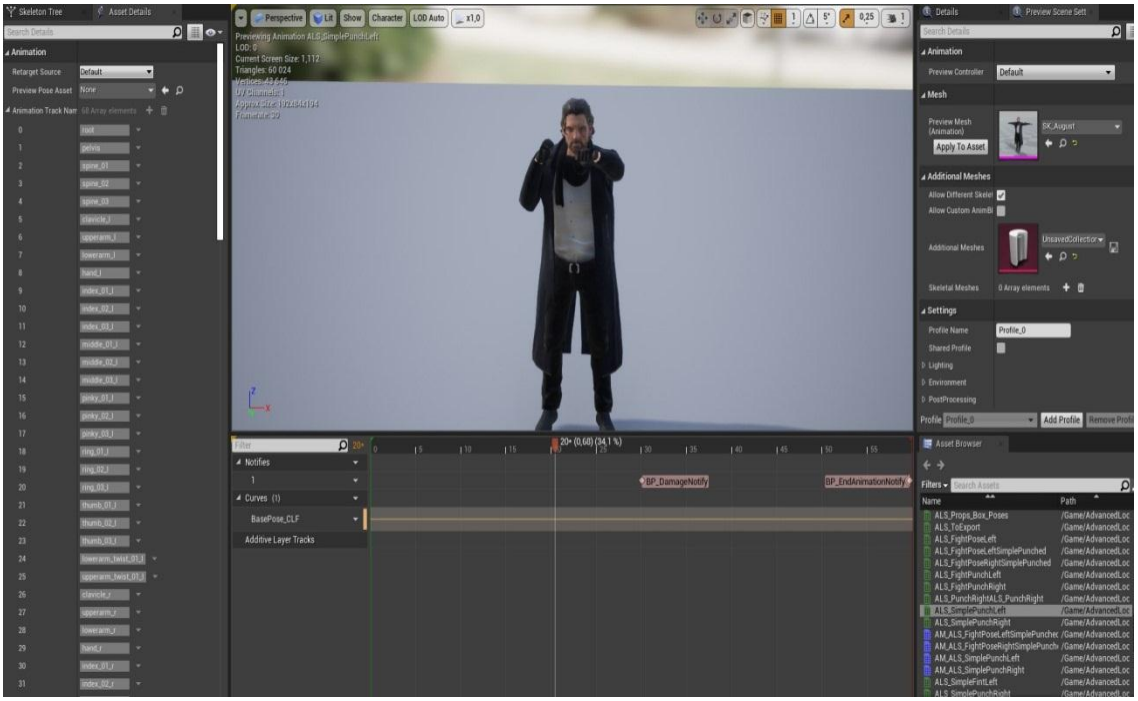


Рисунок 4.6 – Приклад анімації з тригером оберненої дії

У MakeAction робиться наступне:

1). перевіряється - чи можна зробити дію;

2). обирається анімація для дії;

3). повідомляється заданої ШІ (агенту), що гравець почав здійснювати певну дію;

4). починає програватися анімація дії.

Приклад виконання модулю дії представлений на рисунку 4.7.

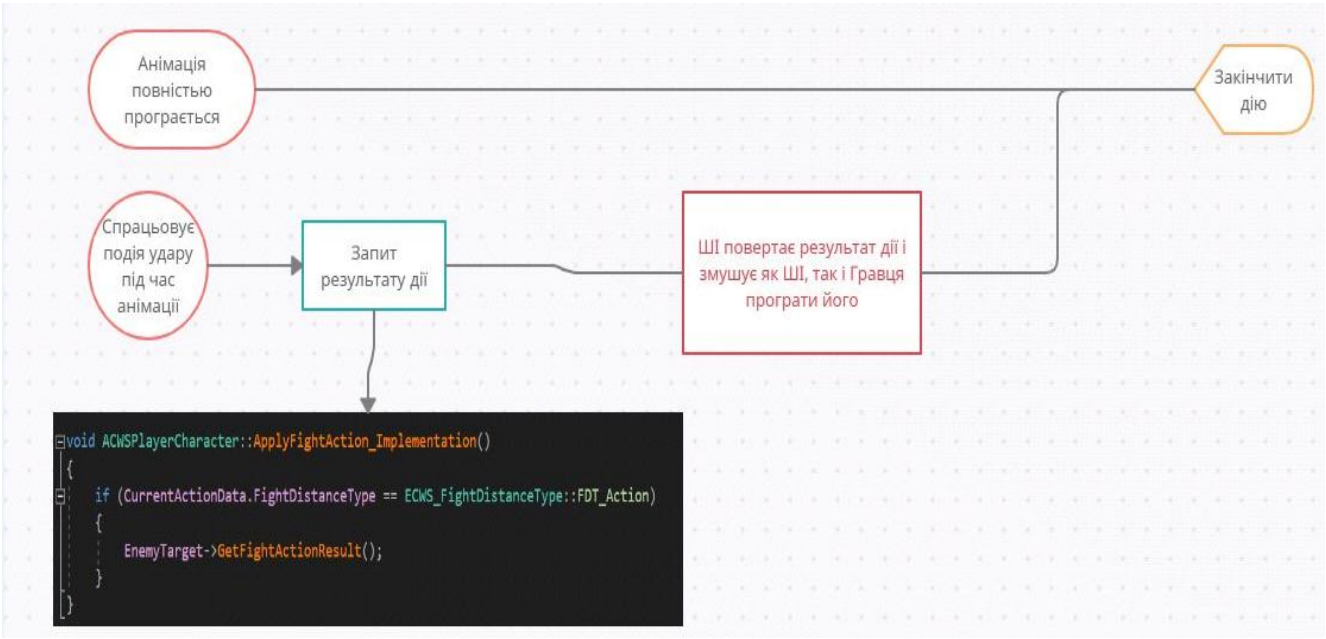


Рисунок 4.7 – Приклад виконання модулю дії

Приклад реалізації протидії агента представлений на рисунку 4.8.

```
if (AIAction == ECWS_FightActionState::FAS_RightBlocking)
{
    // Block
    AIResultData.ResultAnimation = SelectCounterAnimation(PlayerTarget->GetCurrentActionData());
}
else if (AIAction == ECWS_FightActionState::FAS_RightAttacking)
{
    // Calculate who win
    bool bIsWin = PlayerTarget->GetCurrentActionData().StartAnimationTime > CurrentActionData.StartAnimationTime;
}
else
{
    // Hit
    AIResultData.ResultAnimation = SelectCounterAnimation(PlayerTarget->GetCurrentActionData());
}
```

Рисунок 4.8 – Приклад реалізації протидії агента

4.3 Результати роботи ПЗ інтелектуалізованої системи

Важливим питанням є вибір технології, що дає змогу збільшити продуктивність та ефективність використання штучного інтелекту в іграх. Цікавим для практичного використання є технологія компанії Intel призначена для автоматичного збільшення тактової частоти процесора вище номінальної, яка називається Turbo Boost. Але при цьому не повинні перевищуватись обмеження потужності, температури та струму у складі розрахункової потужності. Використання такої технології призводить до збільшення продуктивності одно потокових та багато потокових програм. Іншими словами це технологія фактично саморозгону процесора. Від кількості активних ядер доступність технології Turbo Boost не залежить. Вона залежить від наявності одного або кількох ядер, що мають змогу працювати на потужності яка є нижче розрахункової. Час роботи системи в цій технології залежить від таких речей як робоче навантаження, конструкції платформи та умови експлуатації. Технологія Intel Turbo Boost включена в одному з меню BIOS за замовчуванням.

Ще однією технологією є нейронний процесор або прискорювач штучного інтелекту (NPU). Це спеціалізований клас мікропроцесорів або ще інакше співпроцесорів, що найчастіше використовується для апаратного прискорення роботи штучного інтелекту. Такого як алгоритми штучних нейронних мереж, розпізнавання голосу, комп’ютерного зору, машинного навчання та інших методів штучного інтелекту. Нейронні процесори відносяться до обчислювальної техніки і використовуються для апаратного прискорення емуляції роботи нейронних мереж, а також в режимі реального часу цифрової обробки сигналів. Найчастіше нейропроцесори містять блоки пам’яті магазинного типу, регістри, обчислювальний пристрій, що містить матрицю множення, комутатор, дешифратори, мультиплексори та тригери. До класу нейронних процесорів відносяться:

- Нейроморфні процесори, що побудовані за кластерною архітектурою. На відміну від традиційних обчислювальних архітектур, вони є

вузькоспеціалізованими для створення та розробки різних видів штучних нейронних мереж. В цих процесорах використовуються звичайні транзистори. З них будуються обчислювальні ядра. Кожне ядро містить маршрутизатор, щоб зв’язуватися з іншими ядрами, планувальник завдань та власну пам'ять SRAM. Кожне ядро емулює роботу великої кількості нейронів. Такі процесори застосовуються для глибокого машинного навчання.

- Тензорні процесори, що є співпроцесорами, керуються центральним процесором, що оперують тензорами. Вони мають власну вбудовану оперативну пам'ять і є вузькоспеціалізованими для виконання операцій матричного множення та згортка. Ці операції використовуються для емуляції загорткових нейронних мереж, що використовуються для машинного навчання.

- Процесори машинного зору мають багато схожості з тензорними процесорами. Але відмінні від них тим, що використовуються для того, щоб прискорювати роботу алгоритмів машинного зору, в яких використовуються методи згорткових мереж, а також масштабно-інваріантна трансформація ознак. Акцентування робиться на розпаралелювання потоку даних між множиною виконавчих ядер. Вони також як і тензори використовуються для обчислень з низькою точністю, що прийнята при обробці зображень.

В даній роботі тестування штучного інтелекту було проведено декількома способами: вручну, з використанням простого обрахування процесором, а також з використанням процесора для прискорення штучного інтелекту.

Приклад таблиці з результатами ручного навчання наведений на рисунку 4.5.

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	1.242522	0.992239	-3.471755	-0.971019	-0.514808	0.000000	0.000000
2	State1	0.000000	1.000000	0.000000	0.000000	2.603926	0.000000	0.000000
3	State2	-0.250000	0.000000	2.000000	3.274837	0.000000	0.000000	0.000000
4	State3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	State4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
6	State5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7	State6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Рисунок 4.5 – Таблиця з результатами ручного навчання

Наступне тестування штучного інтелекту за допомогою навчання з підкріпленням було проведено опрацьованням стандартним методом за допомогою сухого обрахування процесором.

У результаті агент навчився взаємодіяти з гравцем лише за 2 години 24 хвилини.

Для повторного тесту було використано процесор Intel I9 11900 (11 покоління), який мав нову архітектуру для штучного інтелекту – Intel Deep Learning Boost.

Приклад зовнішнього вигляду процесора наведено на рисунку 4.6.

Рисунок 4.6 – Приклад використаної моделі процесору для експериментів

З представленої таблиці штучний інтелект обрав перевагу атакувати з лівої руки. В зв’язку з тим, що гравець рідко робить блок правою рукою, то персонаж комп’ютерної гри отримує ушкодження. Що стосується блокування, то він навчився блокувати удари граця після 15 отриманих ударів.

Результати навчання з підкріплення з використанням процесора для прискорення штучного інтелекту наведено на рисунках 4.7 – 4.10.

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.142857	0.285714	0.752381	0.266667	-0.257143	-0.847619	0.333333
2	State1	0.876190	2.135180	2.122956	-0.847619	0.000000	0.790476	0.000000
3	State2	0.476191	0.323810	1.067830	0.000000	1.120466	0.400000	0.000000
4	State3	2.152290	-0.361905	0.961905	-0.742857	0.485714	-1.197333	-0.876190
5	State4	-1.242607	-0.838095	1.109596	0.047619	-1.528828	0.371429	1.861733
6	State5	1.456522	2.128169	-1.876217	1.327616	-4.974322	2.183065	5.696792
7	State6	1.089053	-1.009524	0.828571	0.000000	0.942857	0.000000	0.000000

Рисунок 4.7 – Результати експериментів

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.209524	0.761905	0.466667	-0.114286	-0.247619	0.085714	0.647619
2	State1	0.152381	0.723810	0.428571	-0.028571	0.009524	-0.200000	-0.066667
3	State2	0.171428	0.047619	-0.047619	0.161905	-0.076190	0.380952	-0.104762
4	State3	0.114286	0.266667	0.342857	0.047619	0.152381	-0.095238	-0.019048
5	State4	0.228571	0.161905	-0.161905	0.095238	0.171429	0.047619	-0.114286
6	State5	0.000000	0.466667	0.104762	0.400000	-0.200000	0.285714	0.057143
7	State6	-0.152381	0.161905	0.238095	0.200000	0.152381	0.000000	0.000000

Рисунок 4.8 – Результати експериментів

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.000000	0.495238	0.390476	0.333333	-0.314286	0.000000	0.000000
2	State1	0.000000	0.000000	0.228571	-0.028571	0.247619	-0.085714	0.047619
3	State2	0.000000	0.619048	0.438095	0.000000	0.238095	-0.438095	-0.180952
4	State3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	State4	0.200000	-0.114286	0.266667	-0.238095	0.285714	0.200000	0.133333
6	State5	1.131137	0.885714	0.085714	0.057143	0.400000	0.057143	0.000000
7	State6	0.000000	0.000000	0.000000	0.000000	0.514286	0.314286	0.000000

Рисунок 4.9 – Результати експериментів

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.961905	-0.171429	0.314286	-0.495238	-0.276190	0.238095	0.209524
2	State1	0.266667	0.257143	0.219048	-0.028571	-0.190476	-0.085714	0.180952
3	State2	0.095238	0.238095	0.133333	0.200000	0.171429	-0.152381	-0.161905
4	State3	0.857143	-0.523810	0.466667	0.400000	-0.400000	0.609524	-0.295238
5	State4	0.323810	-0.257143	0.000000	0.342857	-0.514286	0.523810	0.495238
6	State5	-0.171429	0.104762	0.200000	0.152381	0.285714	0.257143	-0.295238
7	State6	-0.161905	-0.314286	0.133333	0.161905	0.209524	0.114286	0.238095

Рисунок 4.10 – Результати експериментів

Проаналізувавши результати навчання штучного інтелекту за допомогою всіх трьох методів, можна зробити висновок, що використання процесору Intel I9 11900 (11 покоління), який мав нову архітектуру для штучного інтелекту – Intel Deep Learning Boost виявилося найбільш ефективним.

Повторне навчання з використанням цього апаратного рішення зайняло лише 1 годину і 7 хвилин, а звільнені ресурси можна було переделегувати на інші процеси.

Ефективність використання запропонованих рішень доведено експериментами. Використання моделі прискорювача штучного інтелекту дозволило в 2,14 рази пришвидшити навчання персонажу коп’ютерної гри порівняно з класичними методами.

4.4 Висновки

З використанням запропонованих рішень розроблено програмне забезпечення інтелектуалізованої системи на основі методів машинного навчання для розроблення комп’ютерних ігор з використанням апаратних прискорювачів штучного інтелекту. Проведено попередній аналіз існуючих інструментальних засобів розроблення ПЗ та існуючих апаратних рішень. Здійснено обгрунтований вибір засобів розроблення ПЗ та апаратних рішень, які є найбільш ефективними для реалізації інтелектуалізованої системи на основі методів машинного навчання для розроблення комп’ютерних ігор.

За допомогою створеного ПЗ інтелектуалізованої системи на основі методів машинного навчання для розроблення комп’ютерних ігор досліджено можливості практичного використання удосконаленого агентно-орієнтованого методу та моделі прискорювача штучного інтелекту.

Ефективність використання запропонованих рішень доведено експериментами. Використання моделі прискорювача штучного інтелекту дозволило в 2,14 рази пришвидшити навчання персонажу коп’ютерної гри порівняно з класичними методами.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень набула подальшого розвитку модель ШІ-прискорювача в сучасних процесорах для обрахунку штучного інтелекту в іграх, для обрахунку штучного інтелекту в іграх, що використовує процесор Intel I9 11900 (11 покоління), а також удосконаленний агентно-орієнтований метод, що дозволяє підвищити продуктивність агента за рахунок врахування вагового коефіцієнта винагороди, що в свою чергу задовільняє функціональну та дизайнерьску систему гри. На основі запропонованого підходу розроблена інтелектуалізована система з використанням процесора Intel I9 11900, що дозволяє при застосуванні методів машинного навчання для розроблення комп'ютерних ігор, підвищити ефективність та швидкість навчання персонажів ігрових проектів.

Практична значимість отриманих результатів полягає у тому, що отримані результати магістерської роботи можуть бути використані для підвищення ефективності використання машинного навчання при розробці комп’ютерних ігор.

У першому розділі роботи був проведений аналіз існуючих інструментальних засобів для полегшення та пришвидшення процесу розробки ігор. Доведено, що використання прискорювачів штучного інтелекту для навчання персонажів комп’ютерних ігор за допомогою інтелектуалізованої системи на основі методів машинного навчання є актуальною задачею. Це дозволяє підвищити ефективність та швидкість обробки алгоритмів штучного інтелекту.

У другому розділі роботи представлена формалізація процесу прийняття рішення з використанням ланцюгів Маркова та модель агента, а також представлений удосконаленний агентно-орієнтований метод, що дозволяє підвищити продуктивність агента за рахунок врахування вагового коефіцієнта винагороди, що в свою чергу задовільняє функціональну та дизайнерьску систему гри. Проведено аналіз існуючих алгоритмів навчання та обгрунтовано переваги

методу машинного навчання з підкріпленням, що доцільно використовувати для навчання агентів у системі. Також досліджено проблеми, які виникають у процесі навчання та способи їх усунення.

У третьому розділі роботи, відповідно до розробленої формалізованої моделі агента та агентно-орієнтованого методу, виконано проектування структури ПЗ та розроблено алгоритм роботи інтелектуалізованої системи на основі методів машинного навчання для розроблення комп'ютерних ігор. Проведений аналіз показав, що для конкретної задачі, що розглядається в даній роботі, доцільним для прискорення навчання ШІ є використання моделі прискорювача штучного інтелекту, що використовує процесор Intel I9 11900 (11 покоління), який мав нову архітектуру для штучного інтелекту – Intel Deep Learning Boost. У процесі проектування структури ПЗ виконано необхідні уточнення структури підсистеми машинного навчання, яка описана у формалізованій моделі, а також проведений аналіз та вибір апартного прискорювача штучного інтелекту.

В четвертому розділі з використанням запропонованих рішень розроблено програмне забезпечення інтелектуалізованої системи на основі методів машинного навчання для розроблення комп'ютерних ігор з використанням апаратних прискорювачів штучного інтелекту. Проведено попередній аналіз існуючих інструментальних засобів розроблення ПЗ та існуючих апаратних рішень. Здійснено обгрунтований вибір засобів розроблення ПЗ та апаратних рішень, які є найбільш ефективними для реалізації інтелектуалізованої системи на основі методів машинного навчання для розроблення комп'ютерних ігор.

За допомогою створеного ПЗ інтелектуалізованої системи на основі методів машинного навчання для розроблення комп'ютерних ігор досліджено можливості практичного використання удосконаленого агентно-орієнтованого методу та моделі прискорювача штучного інтелекту. Ефективність використання запропонованих рішень доведено експериментами. Використання моделі прискорювача штучного інтелекту дозволило в 2,14 рази пришвидшити навчання персонажу комп'ютерної гри порівняно з класичними методами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sze V, Chen Y-H, Yang T-J, Emer J S. Efficient processing of deep neural networks: a tutorial and survey. *Proc IEEE*, 2017. Vol. 105(12). Pp. 2295–2329.

2. Yao X, Zhou J, Zhang J, Boer C R. From intelligent manufacturing to smart manufacturing for industry 4.0 driven by next generation artificial intelligence and further on. In: *5th International Conference on Enterprise Systems (ES)*, 2017. Pp. 311 – 318.

3. Bishnoi L, Narayan Singh S. Artificial intelligence techniques used in medical sciences: a review. In: *8th International Conference on Cloud Computing, Data Science and Engineering (Confluence)*, 2018. Pp. 106–113.

4. Rao Q, Frtunikj J. Deep learning for self-driving cars. In: *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems—SEFAIS ’18*, 2018. Pp. 35–38. doi:10.1145/3194085.3194087.

5. Baji T. Evolution of the GPU device widely used in AI and massive parallel processing. In: *IEEE 2nd Electron Devices Technology and Manufacturing Conference (EDTM)*, 2018. Pp.7–9. doi:10.1109/EDTM.2018.8421507.

6. Shawahna A, Sait SM, El-Maleh A. FPGA-based accelerators of deep learning networks for learning and classification: a review. *IEEE Access*, 2019. Vol. 7. Pp. 7823–7859.

7. Mittal S. A survey of FPGA-based accelerators for convolutional neural networks. *Neural Comput Appl*, 2018. Vol. 32(4). Pp.1109–1139.

8. Apple : (документація) / Classifying Images with Vision and Core ML – 2020. URL: https://developer.apple.com/documentation/vision/classifying_images_with_vision_and_core_ml (дата звернення: 11.02.2021).

9. Jawandhiya P. Hardware design for machine learning. *Int J Artif Intell Appl (IJAIA)*, 2018. Vol. 9(1). Pp. 63–84.

10. Wang T, Wang C, Zhou X, Chen H. A survey of FPGA based deep learning accelerators: challenges and opportunities, *CoRR*, 2018. Vol. 1901.04988.

11. Я. Гудфеллоу, И. Бенджио, А. Курвилль. Глубокое обучение. М.: ДМК Пресс, 2017. 652с.

12. Rigos S. A hardware acceleration unit for face detection. In: *Mediterranean Conference on Embedded Computing (MECO)*, Bar, 2012. Pp. 17–21.

13. Nurvitadhi E, Venkatesh G, Sim J, Marr D, Huang R, Ong Gee Hock J, Liew YT, Srivatsan K, Moss D, Subhaschandra S, Boudoukh G. Can FPGAs beat GPUs in accelerating next-gen- eration deep neural networks? In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays—FPGA '17*, 2017. Pp. 5–14. doi:10.1145/3020078.3021740.

14. Lacey G, Taylor G, Areibi S. Deep learning on FPGAs: past, present, and future, *CoRR*, 2016. Pp. 1–8. arXiv: 1602.04283.

15. Faraone J, Gambardella G, Boland D, Fraser N, Blott M, Leong PHW. Customizing low-precision deep neural networks for FPGAs. In: *28th International Conference on Field Programmable Logic and Applications (FPL), IEEE*, 2018. Pp. 97–102.

16. Cheng Kwang-Ting, Wang Yi-Chu. Using mobile GPU for general-purpose computing; a case study of face recognition on smartphones. In: *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test*, 2011. Pp. 1–4, doi: 10.1109/VDAT.2011.5783575.

17. Ouerhani Y, Jridi M, Al Falou A. Fast face recognition approach using a graphical processing unit "GPU". In: *IEEE International Conference on Imaging Systems and Techniques*, 2010. Pp.80–84.

18. Li E, Wang B, Yang L, Peng Y, Du Y, Zhang Y, Chiu Y-J. GPU and CPU cooperative accel- eration for face detection on modern processors. *Presented at the 2012 IEEE International Confer- ence on Multimedia and Expo (ICME)*, 2012. Pp. 769–775.

19. Shah A. A, Zaidi Z. A, Chowdhry B. S, Daudpoto J. Real time face detection/monitor using raspberry pi and MATLAB. In: *IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, 2016. Pp. 1–4.

20. Oro D, Fernandez C, Saeta J. R, Martorell X, Hernando J. Real-time GPU-based face detec- tion in HD video sequences. In: *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011. Pp. 530–537.

21. Misra J, Saha I. Artificial neural networks in hardware: a survey of two decades of progress. *Neurocomputing*, 2010. Vol. 74(1–3). Pp.239–255.

22. Guo K, Zeng S, Yu J, Wang Y, Yang H. A survey of FPGA-based neural network inference accelerators. *ACM Trans Reconfig Technol Syst*, 2019. Vol. 12(1). Pp. 1–26.

23. Apple : (документація) / Core ML – 2020. URL: https://developer.apple.com/documentation/coreml. (дата звернення: 15.02.2021.)

24. Apple : (документація) / Creating a Model from Tabular Data – 2020. URL: https://developer.apple.com/documentation/createml/creating_a_model_from_tabular_data. (дата звернення: 15.02.2021).

25. Talib, M.A., Majzoub, S., Nasir, Q. et al. A systematic literature review on hardware implementation of artificial intelligence algorithms. *J Supercomput*, 2021. Vol. 77. Pp. 1897–1938. doi:10.1007/s11227-020-03325-8.

26. Heartbeat / Anupam Chugh // Build a SwiftUI + Core ML Emoji Hunt Game for iOS – 2019. URL: https://heartbeat.fritz.ai/build-a-swiftui-core-ml-emoji-hunt-gamefor-ios-eb4465ec4153. (дата звернення: 15.02.2021).

27. Apple : (документація) / Apple Developer Documentation. URL: https://developer.apple.com/documentation/. (дата звернення: 15.02.2021).

28. Carlos Marn-Lora, Miguel Chover, Jos M. Sotoca and Luis A. Garca. A game engine to make games as multi-agent systems, *Advances in Engineering Software*, 2020. Vol. 140, Pp. 102–132.

29. Wu Zong-Han, Kevin Lai, Li-An Lin, Ming-Han Huang and Wen-Kai Tai. Procedurally Generating Game Level with Specified Difficulty, *2018 IEEE Games Entertainment Media Conference (GEM)*, 2018. Pp. 1–9.

30. Nur Rohman Widiyanto. Rpg-stats-generator: Calculate and Generate Player’s and Non-Player Character’s Gameplay Attributes for Role-Playing

Game", *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM), Zenodo*, 2020. Pp.103–-110.

31. Michael Phillips. Experience Graphs: Leveraging Experience in Planning, Ph.D. thesis, Carnegie Mellon University, 2015. P.188.

32. Schroeder B.L., Fraulini N.W., Van Buskirk W.L., Johnson C.I. Using a Non-player Character to Improve Training Outcomes for Submarine Electronic Warfare Operators. In: Sottilare R., Schwarz J. (eds) *Adaptive Instructional Systems. HCII 2020. Lecture Notes in Computer Science*, 2020. Vol 12214. Springer, Cham. Pp.531 -- 542. doi:10.1007/978-3-030-50788-6_39.

33. Krening, S., and M. Feigh, K. Interaction algorithm effect on human experience with reinforcement learning. *ACM Transactions on Human-Robot Interaction*, 2018. Vol. 7. Pp. 1–22.

34. Krening, S. Newtonian action advice: Integrating human verbal instruction with reinforcement learning, *CoRR*, 2018. Pp.720 –-727.

35. Y. Kim, D. Shin, J. Lee, Y. Lee and H. Yoo. 14.3 A 0.55V 1.1mW artificial-intelligence processor with PVT compensation for micro robots, *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, 2016. Pp. 258–-259. doi: 10.1109/ISSCC.2016.7418005.

36. Tracy T., Fu Y., Roy I., Jonas E., Glendenning P. Towards Machine Learning on the Automata Processor. In: Kunkel J., Balaji P., Dongarra J. (eds) *High Performance Computing. ISC High Performance 2016. Lecture Notes in Computer Science*, 2016. Vol 9697. Springer, Cham. Pp. 200–-2018. doi:10.1007/978-3-319-41321-1_11.

37. Nawaf Hazim Barnouti, Sinan Sameer Mahmood Al-Dabbagh, Mustafa Abdul Sahib Naser. Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm. Journal of Computer and Communications, 2016. Vol.4. No.11. Pp.15 –-25.

38. U. A. S. Iskandar, N. M. Diah and M. Ismail. Identifying Artificial Intelligence Pathfinding Algorithms for Platformer Games, *2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 2020. Pp. 74

–-80, doi: 10.1109/I2CACIS49202.2020.9140177.

39. J. Olivito, J. Resano and J. L. Briz. Accelerating Board Games Through Hardware/Software Codesign, in *IEEE Transactions on Computational Intelligence and AI in Games*, Dec. 2017. Vol. 9, no. 4. Pp. 393–-401. doi: 10.1109/TCIAIG.2016.2604923.

40. Nguyen, G., Dlugolinsky, S., Bobák, M. *et al.* Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. *Artif Intell Rev*, 2019. Vol. 52. Pp. 77–124. doi:10.1007/s10462-018-09679-z.

41. P. Andersen, M. Goodwin , O. Granmo. Deep RTS: A Game Environment for Deep Reinforcement Learning in Real-Time Strategy Games, *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 2018. Pp. 1–-8. doi: 10.1109/CIG.2018.8490409).

42. Raschka S., Patterson J., Nolet C. Machine Learning in Python: Main Developments and Technology Trends in Data Science, *Machine Learning, and Artificial Intelligence. Information*, 2020. Vol.11. Pp. 193–-200. doi:10.3390/info11040193.

43. Alexandra Coman, Hector Munoz-Avila. Automated generation of diverse npc-controlling fsms using nondeterministic planning techniques, *in Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2013. Vol. 9, No (1). Pp. 12 –-127.

44. Drake, John. Planning For Non-Player Characters By Learning From Demonstration. *Publicly Accessible Penn Dissertations*, 2018. P. 141. https://repository.upenn.edu/edissertations/2756.

45. D. Babichenko *et al*. The Use of Agent-Based Models As Non-Player Characters in Serious Games, *2020 IEEE 8th International Conference on Serious Games and Applications for Health (SeGAH)*, Vancouver, BC, Canada, 2020. Pp. 1–-8, doi: 10.1109/SeGAH49190.2020.9201889.

46. Miranda M., Sánchez-Ruiz A.A., Peinado F. Building Non-player Character Behaviors By Imitation Using Interactive Case-Based Reasoning. In: *Watson I., Weber R. (eds) Case-Based Reasoning Research and Development.*

ICCBR 2020. Lecture Notes in Computer Science, 2020. Vol. 12311. Pp. 263–278. Springer, Cham. doi: 10.1007/978-3-030-58342-2_17.

47. Frazier, S., Riedl, M. Improving Deep Reinforcement Learning in Minecraft with Action Advice. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *15*(1), 2019. Pp. 146–152. Retrieved from https://ojs.aaai.org/index.php/AIIDE/article/view/5237.

48. M. Xu, H. Shi, Y. Wang. Play games using Reinforcement Learning and Artificial Neural Networks with Experience Replay, *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, Singapore, 2018. Pp. 855 –859. doi: 10.1109/ICIS.2018.8466428.

49. Borovikov Igor, Harder Jesse, Sadovsky Michael, Beirami Ahmad. Towards Interactive Training of Non-Player Characters in Video Games, *2019 ICML Workshop on Human in the Loop Learning (HILL 2019)*, Long Beach, USA, 2019. P. 6.

50. N. R. Widiyanto, S. M. S. Nugroho, M. H. Purnomo. The Calculation of Player's and Non-Player Character's Gameplay Attribute Growth in Role-Playing Game with K-NN and Naive Bayes, *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, Surabaya, Indonesia, 2020. Pp. 103 –110. doi: 10.1109/CENIM51130.2020.9297945.

51. Ramiro A. Agis, Sebastian Gottifredi, Alejandro J. García. An event-driven behavior trees extension to facilitate non-player multi-agent coordination in video games, *Expert Systems with Applications*, 2020.Volume 155. ISSN 0957-4174. doi:10.1016/j.eswa.2020.113457.

52. Kopel M., Hajas T. Implementing AI for Non-player Characters in 3D Video Games. In: Nguyen N., Hoang D., Hong TP., Pham H., Trawiński B. (eds) *Intelligent Information and Database Systems. ACIIDS 2018.* Lecture Notes in Computer Science, 2018. Vol. 10751. Springer, Cham. doi.org:10.1007/978-3-319-75417-8_57.

53. Искусственный интеллект в компьютерных играх. Многоуровневое планирование и реактивное поведение агентов. URL:

http://masters.donntu.edu.ua/2013/fknt/ilkun/library/ii.htm. (дата звернення: 18.03.2021).

54. Могильний С. Б. Машинне навчання з використанням мікрокомп'ютерів: навч.-метод. посіб. / за ред. О. В. Лісового та ін. К., 2019. С. 226.

55. Дмитрієнко В.Д., Заковоротний О.Ю., Носков В.І., Мезенцев М.В. Основи нейрокомп'ютингу: навчально-методичний посібник до практичних занять. Харків: НТМТ, 2014. С. 140.

56. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search, *Nature*, 2016. Vol. 529(7587). Pp. 484–489. doi:10.1038/nature16961.

57. Perez-Liebana D., Samothrakis S., Togelius J., Lucas S., Schaul T. General video game AI: competition, challenges, and opportunities, *AAAI Press*, 2016. Pp. 4335–4337.

58. Crawford G., Muriel D. Video Games as Culture: Considering the Role and Importance of Video Games in Contemporary Society, 2018. Routledge. P. 208. doi:10.4324/9781315622743.

59. Vicki L. Sauter. Decision Support Systems for Business Intelligence John, Wiley & Sons, 2014. P. 480.

60. E.W.T.Ngai, Yong Hu, Y.H.Wong, Yijun Chen, Xin Sun. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 2011. Volume 50, Issue 3. Pp. 559-569. doi: 10.1016/j.dss.2010.08.006.

61. Kenneth W. Goodman. Ethical and Legal Issues in Decision Support Clinical Decision Support Systems. 2016. Pp. 131-146. URL: https://link.springer.com/chapter/10.1007/978-3-319-31913-1_8 (дата звернення: 25.02.2021).

62. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: навчальний посібник. Запоріжжя: ЗНТУ, 2010. С. 341.

63. Gemp I. Automated Data Cleansing through Meta-learning / Gemp I., Theocharous G., Ghavamzadeh M., Association for the Advancement of Artificial Intelligence, 2017. URL: https://people.cs.umass.edu/imgemp/pubs/iaai_2017.pdf. (дата звернення: 12.03.2021).

64. Bresnick J. How to Choose the Right Healthcare Big Data Analytics Tools / Bresnick J. URL: https://healthitanalytics.com/features/how-to-choose-the-right-healthcarebig-data-analytics-tools. (дата звернення: 12.03.2021).

65. Bresnick J. IBM Watson Expands Role in Imaging Analytics, Population Health / Bresnick J. URL: https://healthitanalytics.com/news/ibm-watson-expands-role-in-imaginganalytics-population-health. (дата звернення: 12.03.2021).

66. Bresnick J. Google's Machine Learning, Imaging Analytics Flag Breast Cancer / Bresnick J. URL: https://healthitanalytics.com/news/googles-machine-learning-imaginganalytics-flag-breast-cancer. (дата звернення: 12.03.2021).

67. Bresnick J. Data Governance Key to Hospital's Natural Language Query Project / Bresnick J. URL: https://healthitanalytics.com/news/data-governance-key-to-hospitalsnatural-language-query-project. (дата звернення: 12.03.2021).

68. Bresnick J. Machine Learning, NLP Help with Physician Skill Benchmarking. URL: https://healthitanalytics.com/news/machine-learning-nlp-help-withphysician-skill-benchmarking. (дата звернення: 12.03.2021).

69. Bresnick J. Mount Sinai Uses Machine Learning for Heart Imaging Analytics. URL: https://healthitanalytics.com/news/mount-sinai-uses-machine-learning-forheart-imaging-analytics. (дата звернення: 15.03.2021).

70. Bresnick J. UCSF to Develop Machine Learning for CDS, Imaging Analytics. URL: https://healthitanalytics.com/news/ucsf-to-develop-machine-learning-forcds-imaging-analytics. (дата звернення: 15.03.2021).

71. Rayan A. Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks / Payan A., Montana G. URL: https://arxiv.org/abs/1502.02506. (дата звернення: 15.03.2021).

72. Rodrıguez J.J. Rotation Forest: A New Classifier Ensemble Method / Rodrıguez J.J., Kuncheva L.I., Alonso C.J. // IEEE Transactions on Pattern Analysis

and Machine Intelligence. URL: https://ieeexplore.ieee.org/abstract/document/1677518/. (дата звернення: 15.03.2021).

73. Дьяконов А. Введение в анализ данных и машинное обучение. URL: https://alexanderdyakonov.files.wordpress.com/2017/06/book_boosting_pd f.pdf 29. (дата звернення: 15.03.2021).

74. Chen T. XGBoost: A Scalable Tree Boosting System/ Chen T., Guestrin C. URL: https://arxiv.org/abs/1603.02754. (дата звернення: 15.05.2021).

75. Tumer K.A Error Correlation and Error Reduction in Ensemble Classiers. URL: https://www.tandfonline.com/doi/abs/10.1080/09540099611683931. (дата звернення: 15.03.2021).

76. Ali S. Can–Evo–Ens: Classifier stacking based evolutionary ensemble system for prediction of human breast cancer using amino acid. URL: https://www.sciencedirect.com/science/article/pii/S1532046415000064?via %3. (дата звернення: 15.03.2021).

77. Довгаль Д.О., Жданова О.Г. Підтримка прийняття рішень в слабкостуктурованих системах: Матеріали III всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019), м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 20-22 листопада 2019 р. С. 120-125.

78. Shulman. I. Medical Information Systems "axiom of usability". Available at. URL: http://www.pcweek.ru/themes/detaiI. php?ID=73462. (дата звернення: 15.03.2021).

79. Lia, C.-M., Duc, Y.-C., Wua, J.-X., Lind, C.-H., Hoe, Y.-R., Lina, Y., Chen, T. Synchronizing chaotification with support vector machine and wolf pack search algorithm for estimation of peripheral vascular occlusion in diabetes mellitus. *Biomedical Signal Processing and Control*, 2014. Vol. 9. Pp. 45-55.

80. Krizhevsky A., Sutskever I., Hinton G.. Imagenet classification with deep convolutional neural networks: *Advances in Neural Information Processing Systems*, 2012. Vol. 1. Pp. 1097–1105.

81. S. Nitish, G. Hinton, A. Krizhevsky. Dropout: A Simple Way to Prevent Neural Networks from overfitting: *Journal of Machine Learning Research*, 2014. Vol. 15. Pp. 1929–1958.

82. Salamon J. A Dataset and Taxonomy for Urban Sound Research / J. Salamon, C. Jacoby, J. Bello. // 22nd ACM International Conference on Multimedia, Orlando USA. 2014.

83. Learning Deep Features for Discriminative Localization / B. Zhou, A. Khosla, A. Lapedriza та ін. 2015. URL: https://arxiv.org/abs/1512.04150. (дата звернення 08.04.2021).

84. Visual Explanations from Deep Networks via Gradient-based Localization / R. Selvaraju, M. Cogswell, A. Das та ін. 2017. URL: https://arxiv.org/abs/1610.02391. (дата звернення 08.04.2021).

85. Edward H. Shortliffe, Martin J. Sepúlveda. Clinical Decision Support in the Era of Artificial Intelligence JAMA. 2018. Pp. 2199-2200. doi: https://doi.org/10.1001/jama.2018.17163.

86. Decision Support Systems VIII: Sustainable Data-Driven and Evidence-Based / Fatima Dargam, Pavlos Delias, Isabelle Linden, Bertrand Mareschal. 2018. doi: https://doi.org/10.1007/978-3-319-90315-6.

87. Николенко С. И. Глубокое обучение. Погружение в мир нейронных сетей. / С. И. Николенко, А. А. Кадурин, Е. О. Архангельская., 2018. С. 480.

88. Simonyan K. Very Deep Convolutional Networks for Large-Scal Image Recognition / K. Simonyan. 2014. URL: http://arxiv.org/abs/1409.1556. (дата звернення: 24.04.2021).

89. Gwardys G. Deep image features in music information retrieval / G. Gwardys, D. Grzywczak. // International Journal of Electronics and Telecommunications. 2014. Vol. 60. Pp. 321–326.

90. Striving for Simplicity: The All Convolutional Net / J.Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller. 2014. URL: https://arxiv.org/abs/1412.6806. (дата звернення 20.04.2021).

91. Dropout: A Simple Way to Prevent Neural Networks from overfitting / S. Nitish, G. Hinton, A. Krizhevsky / Journal of Machine Learning Research, 2014. Vol. 15. С. 1929–1958. URL: http://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf?source=post_page. (дата звернення 28.03.2021).

92. Visual Explanations from Deep Networks via Gradient-based Localization / R. Selvaraju, M. Cogswell, A. Das та ін. 2017. URL: https://arxiv.org/abs/1610.02391. (дата звернення 28.03.2021).

93. Learning Deep Features for Discriminative Localization / B. Zhou, A. Khosla, A. Lapedriza. 2015. URL: https://arxiv.org/abs/1512.04150. (дата звернення 28.03.2021).

94. Лямец В.И., Успенко В.И. Основы общей теории систем и системный анализ. Учебное пособие : Харьков: «БУРУН и К», Киев: ООО «КНТ», 2015. С. 304.

95. Ye. Hnatchuk, Yev. Sierhieiev, A. Hnatchuk. Using artificial intelligence accelerators to train computer game characters, 2021, P. 11 (подано в журнал Computer Systems and Information Technologies).

ДОДАТОК А

(обов’язковий)

Фрагменти коду програмного забезпечення

```
void
ACWSPlayerCharacter::MakeFightAction_Implementation(ECWS_FightActionType
ActionType)
{
	bIsMakeActionCanContinue = true;
	if (!isCanMakeAction)
	{
		NextFightAction = ActionType;
		bIsMakeActionCanContinue = false;
		return;
	}

	if(CurrentActionData.FightActionState !=
ECWS_FightActionState::FAS_Standing)
	{
		UE_LOG(LogTemp, Error, TEXT("FightActionState should be
Standing"))
		bIsMakeActionCanContinue = false;
		return;
	}

	EnemyTarget = Cast<ACWSAICharacter>(TargetCameraComponent-
>GetTargetActor());
```

```
	NextFightAction = ECWS_FightActionType::FAT_None;

	GetController()->SetIgnoreMoveInput(true);

	CurrentActionData.AnimationParam = SelectActionAnimation(ActionType);

	EnemyTarget->NotifyPlayerAction(this, ActionType);

}
```

Функція початку дії агента у коді

Розширена функція дії у блупринтах.

```
void ACWSPlayerCharacter::GetFightActionResult_Implementation()
{
	isHaveCompareResult = true;
	EnemyTarget->SetIsHaveCompareResult(true);

	const auto PlayerAction = CurrentActionData.FightActionState;
	const auto AIAction = EnemyTarget->GetCurrentActionData().FightActionState;

	FCWSFightActionResultData PlayerResultData;
	PlayerResultData.bIsCauser = false;
```

```
FCWSFightActionResultData AIResultData;
AIResultData.bIsCauser = true;

switch (AIAction)
{
    case ECWS_FightActionState::FAS_LeftAttacking:
    {
        if (PlayerAction == ECWS_FightActionState::FAS_RightBlocking)
        {
            // Block
            PlayerResultData.ResultAnimation =
SelectCounterAnimation(EnemyTarget->GetCurrentActionData());
        }
        else if (PlayerAction ==
ECWS_FightActionState::FAS_RightAttacking)
        {
            // Calculate who win
            bool bIsWin = EnemyTarget-
>GetCurrentActionData().StartAnimationTime >
CurrentActionData.StartAnimationTime;
        }
        else
        {
            // Hit
            PlayerResultData.ResultAnimation =
SelectCounterAnimation(EnemyTarget->GetCurrentActionData());
        }
```

```
		}
		break;
		case ECWS_FightActionState::FAS_RightAttacking:
		{
			if (PlayerAction == ECWS_FightActionState::FAS_RightBlocking)
			{
				// Block
				PlayerResultData.ResultAnimation =
SelectCounterAnimation(EnemyTarget->GetCurrentActionData());
			}
			else if (PlayerAction ==
ECWS_FightActionState::FAS_RightAttacking)
			{
				// Calculate who win
				bool bIsWin = EnemyTarget-
>GetCurrentActionData().StartAnimationTime >
CurrentActionData.StartAnimationTime;
			}
			else
			{
				// Hit
				PlayerResultData.ResultAnimation =
SelectCounterAnimation(EnemyTarget->GetCurrentActionData());
			}
		}
		break;
		case ECWS_FightActionState::FAS_GrabRush:
```

```
            {


            }
            break;
        }

        this->PlayFightActionResult(PlayerResultData);
        EnemyTarget->PlayFightActionResult(AIResultData);
}

void ACWSAIEnemyController::StartCloseCombat()
{
        bIsCloseCombatActive = true;
        GetWorldTimerManager().SetTimer(CombatActionLoop_TimerHandle, this,
&ACWSAIEnemyController::MakeAIAction, 2.f);
        MachineLearningComponent->ReadFromDataTable();
}
```

Початок циклу QTable з інтервалом в 2 секунди

```
void ACWSAIEnemyController::NotifyPlayerAction(ECWS_FightActionType
ActionType)
{
        GetWorldTimerManager().ClearTimer(CombatActionLoop_TimerHandle);


        MakeCounterAIAction(ActionType);
```

```
	UE_LOG(LogTemp, Warning, TEXT("NotifyPlayerAction"));
}
```

Якщо гравець здійснює дію – таймер переривається и одразу іде обрахунок контр-дії.

```
ECWS_FightActionType
UCWSMachineLearningComponent::GetAction(ECWS_FightActionState ActionState)
{
	if (FMath::RandRange(0.f, 1.f) < RandomActionRate)
	{
		//random
		return static_cast<ECWS_FightActionType>(FMath::RandRange(0, 6));
	}
	else
	{
		return GetSmartAction(ActionState);
	}
	return ECWS_FightActionType::FAT_None;
}
```

Штучний інтелект обирає яку зробити дію(випадкову або обдуманну) за допомогою випадкового числа.

```
ECWS_FightActionType
UCWSMachineLearningComponent::GetSmartAction(ECWS_FightActionState
ActionState)
```

```
{
	Actions CurActions = QTable[static_cast<uint8>(ActionState)];

	int32 BestAction = 0;

	float max = 0;
	for (int i = 0; i < CurActions.Num(); ++i)
	{
		if (CurActions[i] > max)
		{
			BestAction = i;
		}
	}

	CurrentState = static_cast<uint8>(ActionState);
	CurrentAction = BestAction;

	return static_cast<ECWS_FightActionType>(BestAction);
}
using UnrealBuildTool;
using System.Collections.Generic;
public class CityWithoutSunTarget : TargetRules
{
	public CityWithoutSunTarget( TargetInfo Target) : base(Target)
	{
```

```
        Type = TargetType.Game;
        DefaultBuildSettings = BuildSettingsVersion.V2;
        ExtraModuleNames.AddRange( new string[] { "CityWithoutSun" } );
    }
}
```

Обирається найоптимальніша дія

```
float UCWSMachineLearningComponent::GetReward(FCWSFightActionResultData ResultData)
{
    switch (ResultData.ResultActionType)
    {
    case ECWS_ResultType::RT_Nothing: return 0.f;
    case ECWS_ResultType::RT_ReceivedDamage:  return -25.f;
    case ECWS_ResultType::RT_AppliedDamage:  return 25.f;
    case ECWS_ResultType::RT_BlockedHit:  return 5.f;
    case ECWS_ResultType::RT_HitBlock: return -5.f;
    default: return 0.f;
    }
}
```

Бали за результат дії

```
void UCWSMachineLearningComponent::UpdateQTable(FCWSFightActionResultData ResultData)
{
    OldAction = CurrentAction;
```

```
        OldState = CurrentState;
        CurrentState = 0;
        float DeltaError = GetReward(ResultData) + gamma *
GetMaxQAction(CurrentState) - QTable[OldState][OldAction];
        QTable[OldState][OldAction] = QTable[OldState][OldAction] + (alpha*
DeltaError);
        SaveToDataTable();
}
```

Оновлення таблиці після результату дії

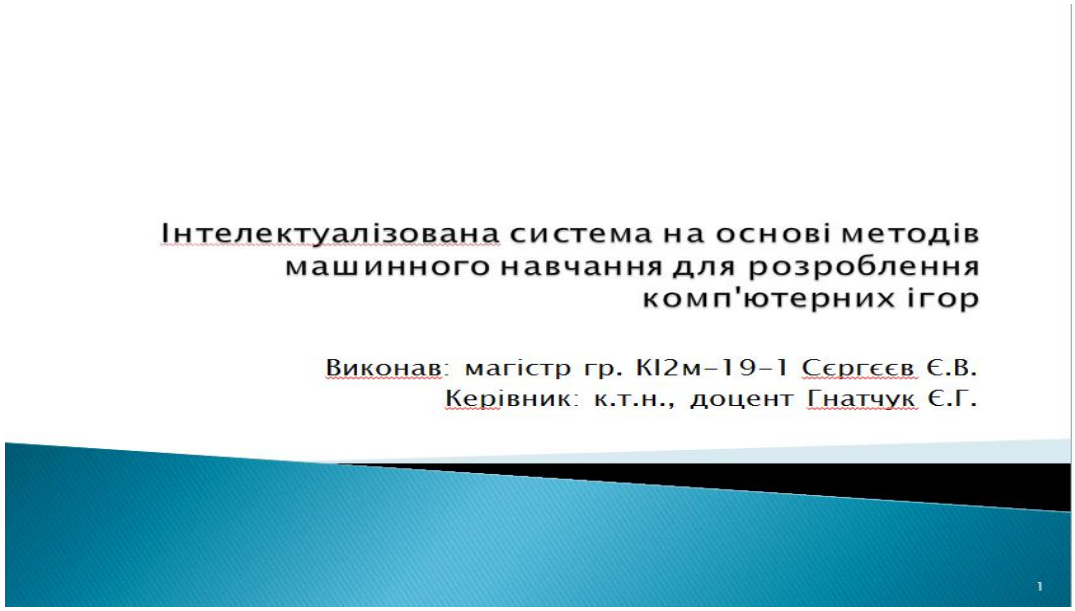
```
using UnrealBuildTool;
using System.Collections.Generic;


public class CityWithoutSunEditorTarget : TargetRules
{
        public CityWithoutSunEditorTarget( TargetInfo Target) : base(Target)
        {
                Type = TargetType.Editor;
                DefaultBuildSettings = BuildSettingsVersion.V2;
                ExtraModuleNames.AddRange( new string[] { "CityWithoutSun" } );
        }
}
```

ДОДАТОК Б

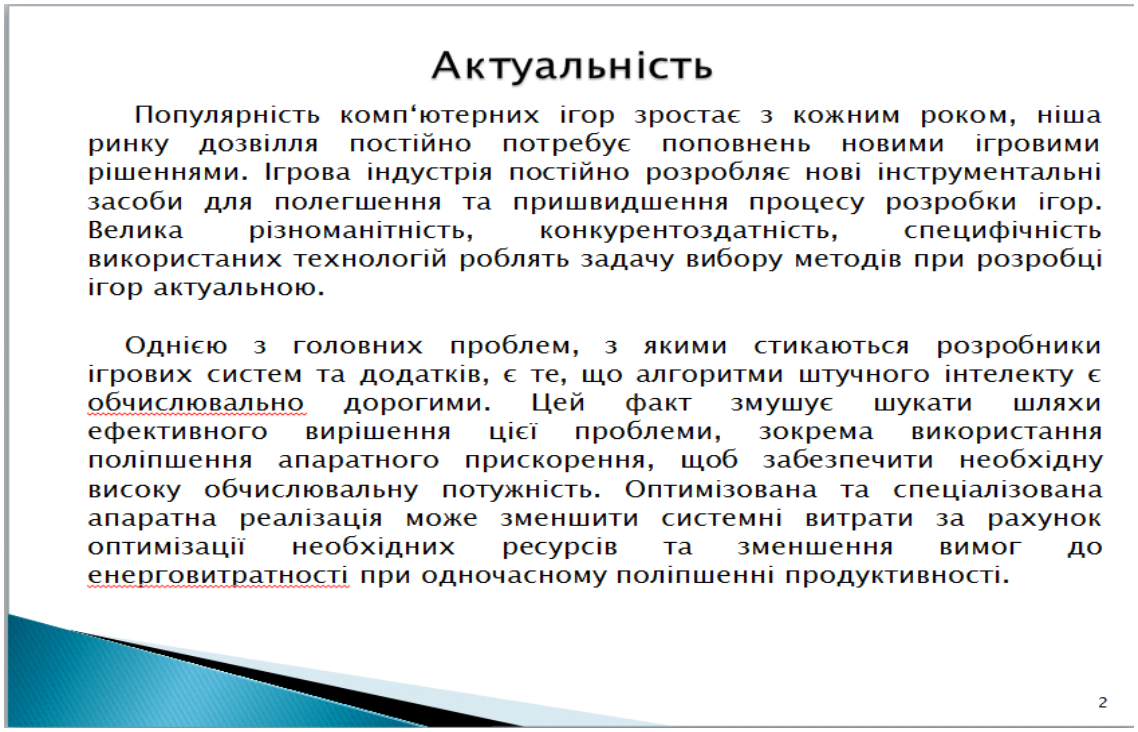
(обов’язковий)

ПРЕЗЕНТАЦІЯ

СЛАЙД 1



СЛАЙД 2



СЛАЙД 3

- **Об'єкт дослідження** – процес прискорення обрахунку штучного інтелекту під час машинного навчання при розробці комп'ютерних ігор.
- **Предмет дослідження** – інтелектуалізований підхід з використанням апаратних засобів на основі методів машинного навчання для розроблення комп'ютерних ігор.
- **Задачі дослідження**:
- 1) провести огляд існуючих апаратних та програмних технологій, що використовуються для підвищення ефективності штучного інтелекту під час машинного навчання;
- 2) провести огляд існуючих методів машинного навчання при розробці комп'ютерних ігор;
- 3) запропонувати технологію вибору апаратного та програмного забезпечення методів машинного навчання при розробці комп'ютерних ігор;
- 4) на основі запропонованої технології розробити програмні засоби.

3

СЛАЙД 4

- **Наукова новизна** отриманих результатів:
- 1) Набула подальшого розвитку модель ШІ-прискорювача в сучасних процесорах для обрахунку штучного інтелекту в іграх, що використовує процесор Intel I9 11900 (11 покоління).
- 2) Удосконалений агентно-орієнтований метод, що дозволяє підвищити продуктивність агента за рахунок врахування вагового коефіцієнта винагороди. що в свою чергу задовольняє функціональну та дизайнерську систему гри.
- На основі запропонованого підходу розроблена інтелектуалізована система з використанням процесора Intel I9 11900 (11 покоління), що дозволяє при застосуванні методів машинного навчання для розроблення комп'ютерних ігор, підвищити ефективність та швидкість навчання персонажів ігрових проектів.
- Практична значимість отриманих результатів полягає у тому, що отримані результати магістерської роботи можуть бути використані для підвищення ефективності використання машинного навчання при розробці комп'ютерних ігор.
- **Зв'язок роботи з науковими програмами, планами, темами.** Дослідження, представлені у кваліфікаційній роботі, проводились в рамках держбюджетної НДР Хмельницького національного університету № 1Б-2019 «Агентно-орієнтована система підвищення безпеки та якості програмного забезпечення комп'ютерних систем» (номер державної реєстрації 0119U100662).
- За темою дипломної роботи подано статтю «Using artificial intelligence accelerators to train computer game characters» (подано в журнал Computer Systems and Information Technologies).

4

СЛАЙД 5

Аналіз останніх досліджень та публікацій

Рік публікації	Дослідження	Опис та сфера застосування
2010	Штучні нейронні мережі в апаратному забезпеченні: огляд двох десятиліть розвитку	У цій оглядовій роботі розглядаються всі основні підходи та моделі апаратних нейронних мереж протягом 1990–2010 років. Вони надають приклади різноманітних реалізацій АНМ у широкому діапазоні моделей ШНМ, таких як згорткові нейронні мережі, нейронна мережа, що підключається, тощо. Ці АНМ реалізації цифрові, аналогові, гібридні, нейроморфні, ПЛІС або оптичні. Кожен з них обговорюється окремо в різних розділах
2013	Огляд використання програмного та апаратного забезпечення у штучних нейронних мережах	Дослідження полягало в тому, чи використовують розробники готові відкриті джерела програмного забезпечення та апаратних прискорювачів або розробляти власні власні версії. Також представлена таблиця програмних плат, що використовуються для штучних нейронних мереж
2017	Огляд прискорювача нейронних мереж на базі FPGA	Це дослідження дає огляд попередньої роботи над нейронною мережеюприскорювачів висновків на основі ПЛІС та узагальнюють основні використовувані методи
2017	Ефективна обробка глибинних нейронних мереж: підручник та огляд	Ця стаття зосереджується лише на типі ANN глибоких нейронних мереж. У ньому згадується історія, компоненти та програми із апаратним забезпеченням, яке підтримує операції типу DNN
2018	Огляд прискорювачів на основі FPGA для згорткових нейронних мереж	Обговорюються кілька стратегій оптимізації, що використовуються в апаратних архітектурах для CNN, такі як переупорядкування циклу, розгортання та конверизація, формат з фіксованою точкою тощо. Автори також класифікують усі реалізації відповідно до підходів оптимізації для архітектури CNN та FPGA або оптимізації пам'яті
2018	Еволюція пристрою графічного процесора, який широко використовується в ШІ та масивній паралельній обробці	Стаття представляє коротку дискусію про історію процесорів та графічних процесорів, детально описуючи закон Мура. Він також містить кілька важливих програм для ШІ

Рік публікації	Дослідження	Опис та сфера застосування
2018	Опитування прискорювачів глибокого навчання на основі FPGA: виклики та можливості	У статті представлено опис конструкцій прискорювачів глибокого навчання на базі FPGA. Вони класифікують роботу відповідно до конструкції або для певного алгоритму, або для конкретного застосування, або для універсального прискорювача фреймворк із апаратними шаблонами
2019	Прискорювачі мереж глибокого навчання на основі FPGA для навчання та класифікації: огляд	У статті дається вичерпна довідка та історія прискорювачів на основі FPGA. Огляд включає програми глибокого навчання та обговорює приклади реалізації графічного процесора, ASIC (з конкретними прикладами) та прикладів ПЛІС у детальному аналізі та тестуванні

5

СЛАЙД 6

- Враховуючи складність обчислювальних процесів та велику вартість цих процесів, ігрова комп'ютерна галузь потребує вдосконалення апаратного та програмного забезпечення для підвищення ефективності та швидкості обробки алгоритмів штучного інтелекту.
- Отже, використання прискорювачів штучного інтелекту для навчання персонажів комп'ютерних ігор за допомогою інтелектуалізованої системи на основі методів машинного навчання є актуальною задачею.
- В процесі виконання роботи необхідно розв'язати наступні задачі:
- 1) провести огляд існуючих апаратних та програмних технологій, що використовуються для підвищення ефективності штучного інтелекту під час машинного навчання;
- 2) провести огляд існуючих методів машинного навчання при розробці комп'ютерних ігор;
- 3) запропонувати технологію вибору апаратного та програмного забезпечення методів машинного навчання при розробці комп'ютерних ігор;
- 4) на основі запропонованої технології розробити програмні засоби.

6

СЛАЙД 7

Агентна модель навчання з підкріпленням

- Агентна модель навчання з підкріпленням

$$AGP = (Zs, St, Rt, Ag, Vg, Wr), \quad (1)$$

- де *Zs* – зовнішнє середовище;
- *St* – стан;
- *Rt* – винагорода;
- *Ag* – набір дій агента;
- *Vg* – набір дій суперника;
- *Wr* – ваговий коефіцієнт винагороди.
- Агент взаємодіє з зовнішнім середовищем, переходить з стану в стан та отримує винагороду за правильно виконану дію. В нашій роботі агент змінює свій стан в залежності від впливу зовнішнього середовища. Множина станів агента представлена формулою 2.2.

$$St = (st_0, st_1, st_2, st_3, st_4, st_5, st_6, st_7), \quad (2)$$

- де st_0 – початковий стан агента;
- $st_1 \ldots st_7$ – стани агента в залежності від впливу зовнішнього середовища.

7

СЛАЙД 8

Агентна модель навчання з підкріпленням

- Множина вагових коефіцієнтів винагороди агента представлена формулою 3.

$$Wr = (wr_1, wr_2, wr_3, wr_4, wr_5), \quad (3)$$

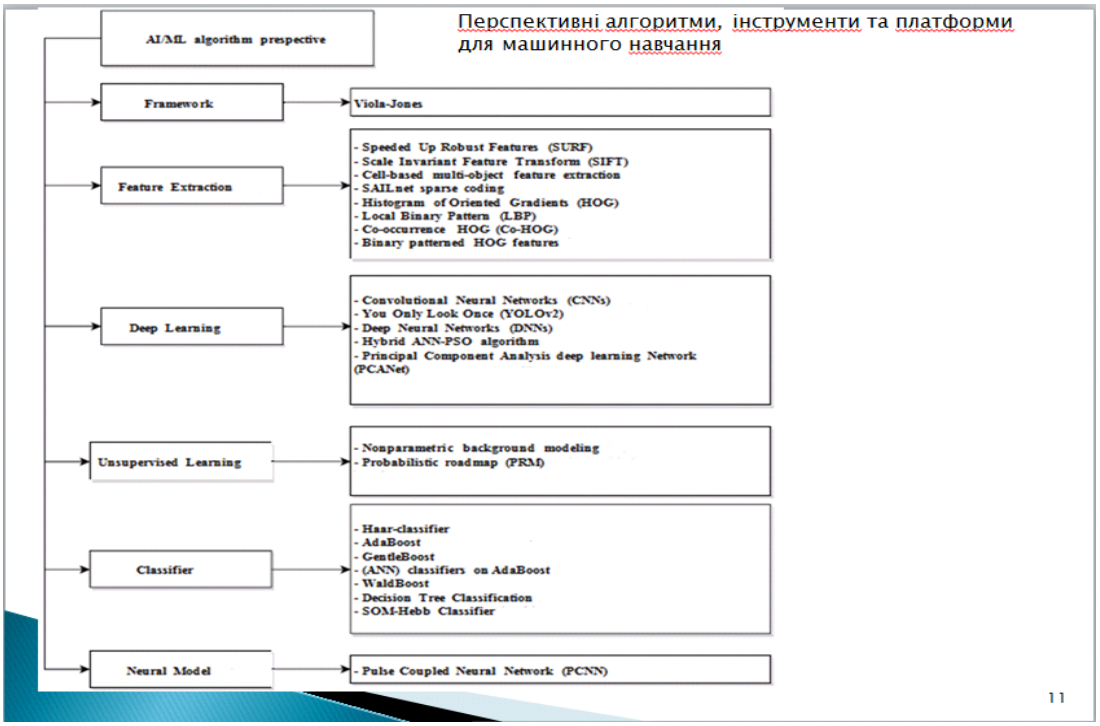
- Вагових коефіцієнтів є п'ять, розподіляються вони наступним чином. Відбуваються певні дії, в залежності від зовнішнього середовища, за які агент отримує винагороду.
- На вхід подається результат дії, якщо нічого не відбулось – функція повертає – 0 очок, якщо агент отримав шкоди, то функція повертає –25 очок, якщо ж агент наніс шкоди, то 25 очок, при блокуванні удару повертається 5 очок і відповідно, якщо агент вдарив по блоку – –5 очок.

8

СЛАЙД 9

СЛАЙД 10

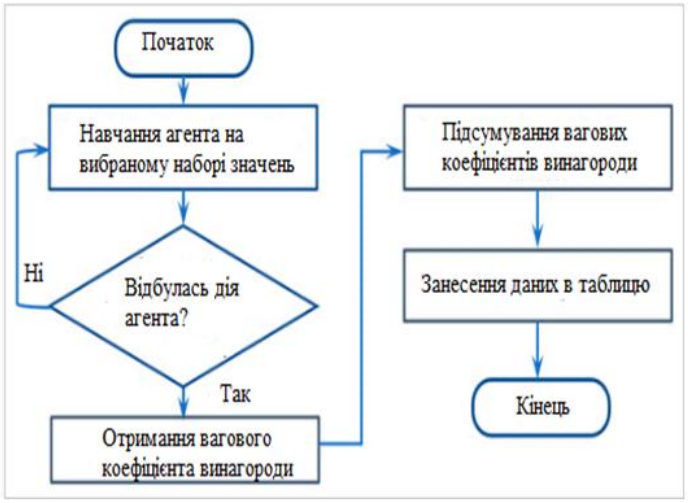
СЛАЙД 11



СЛАЙД 12

СЛАЙД 13

Алгоритм агентно-орієнтованого методу



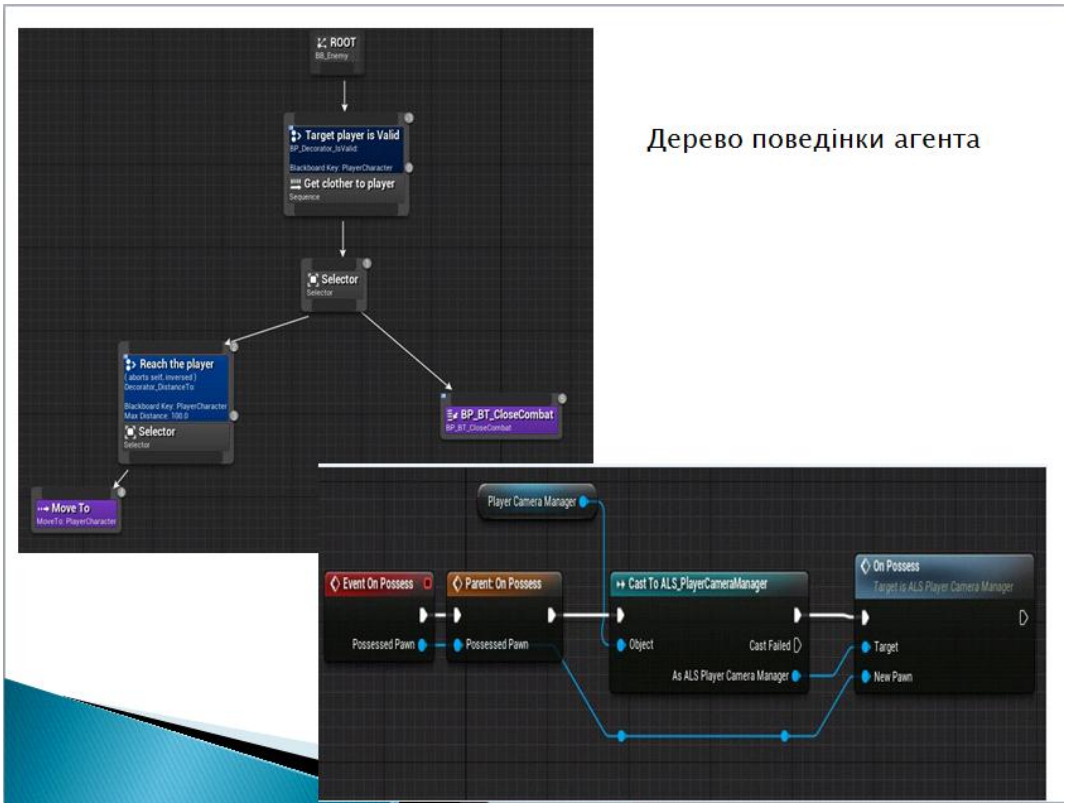
13

СЛАЙД 14

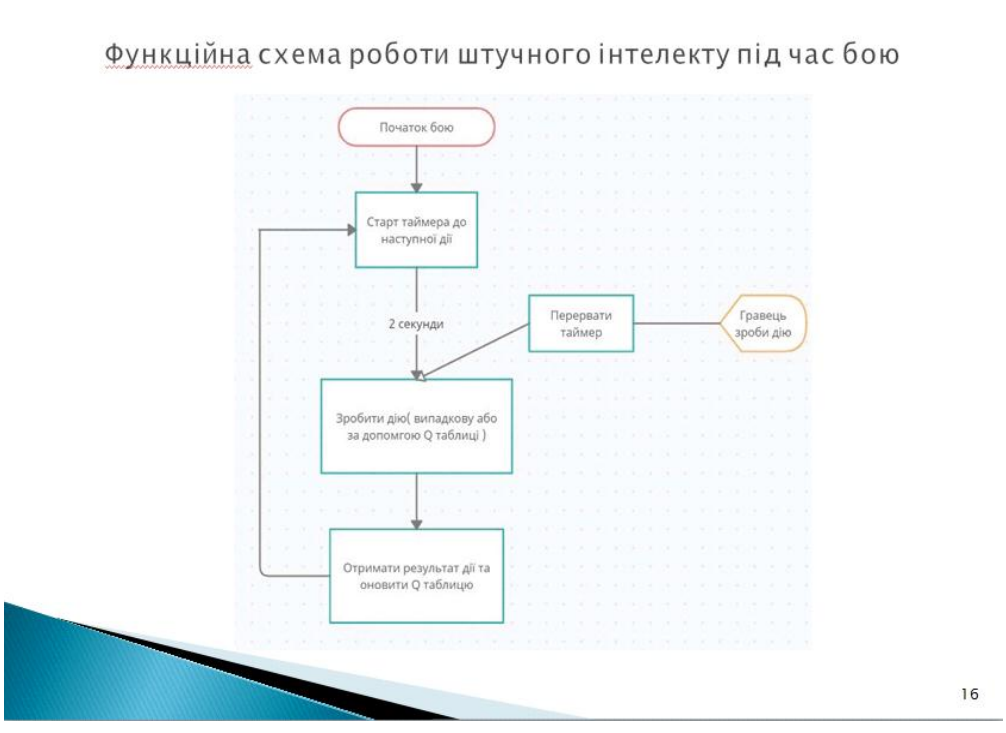
Функційна схема, що описує процес навчання агента з використанням агентно-орієнтованого методу

14

СЛАЙД 15

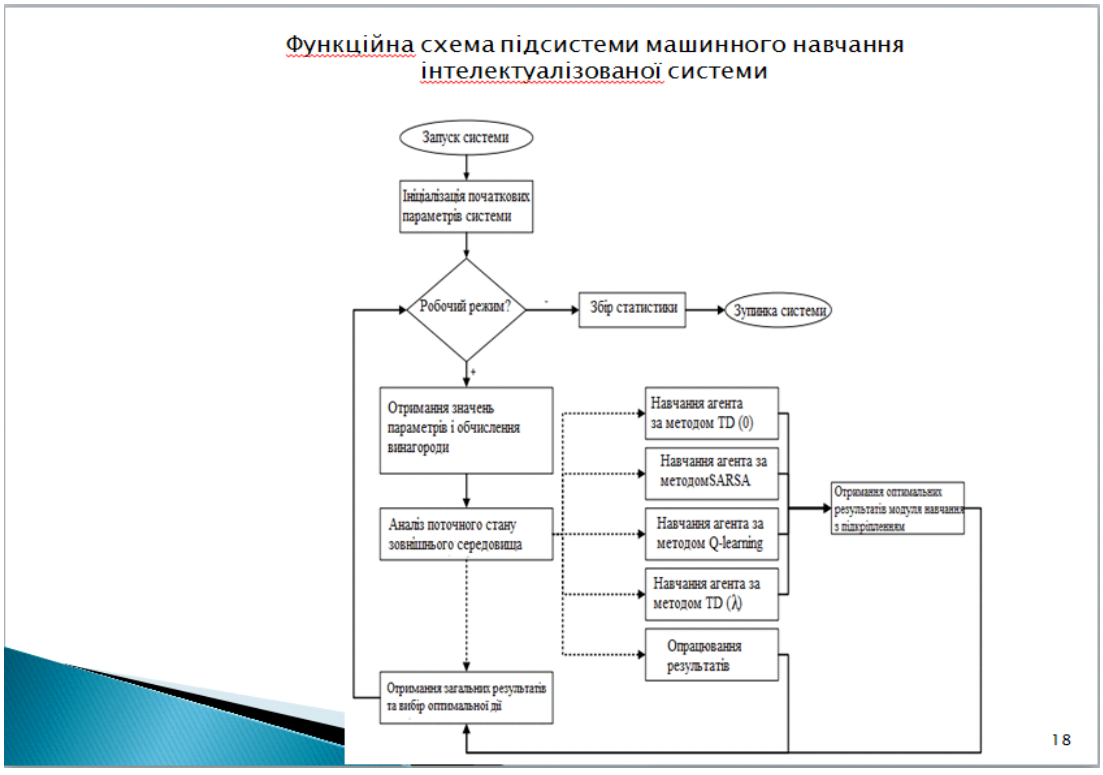


СЛАЙД 16



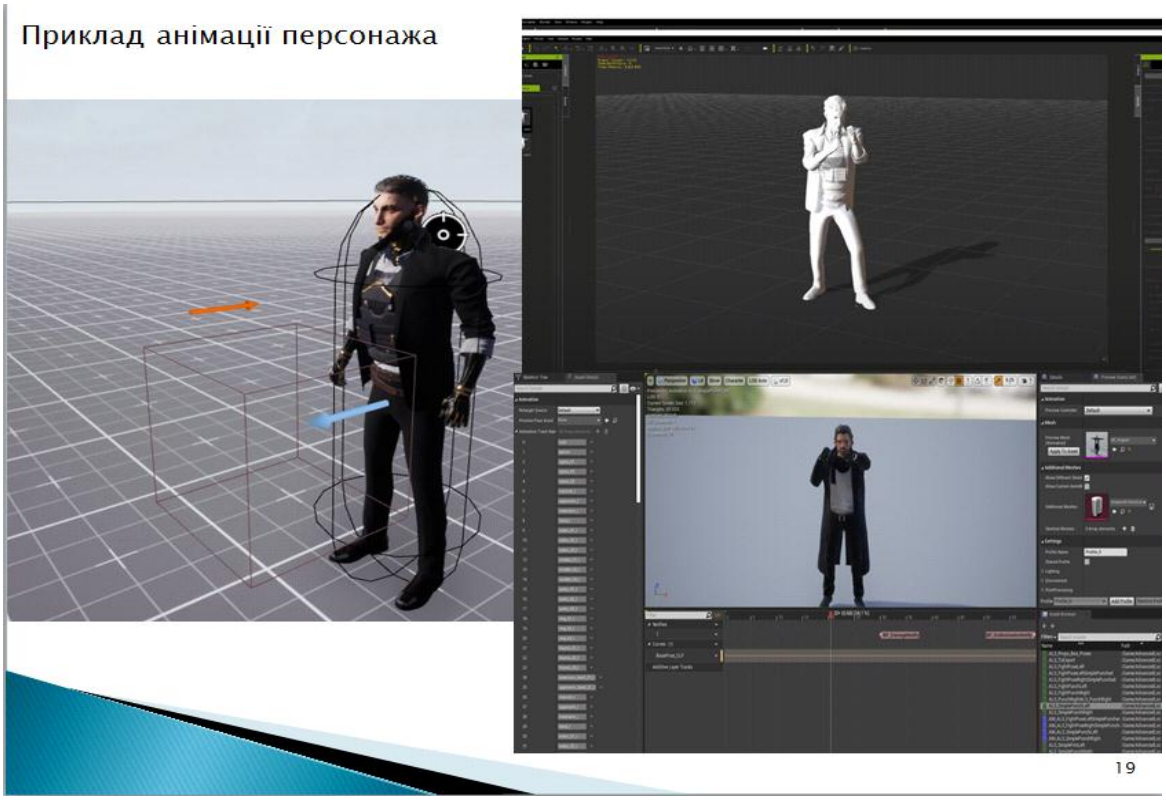
СЛАЙД 17

СЛАЙД 18



СЛАЙД 19

Приклад анімації персонажа



19

СЛАЙД 20

Таблиця з результатами ручного навчання

	Row N	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	1.242522	0.992239	-3.471755	-0.971019	-0.514808	0.000000	0.000000
2	State1	0.000000	1.000000	0.000000	0.000000	2.603926	0.000000	0.000000
3	State2	-0.250000	0.000000	2.000000	3.274837	0.000000	0.000000	0.000000
4	State3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	State4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
6	State5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7	State6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Наступне тестування штучного інтелекту за допомогою навчання з підкріпленням було проведено опрацьованням стандартним методом за допомогою сухого обрахування процесором.
У результаті агент навчився взаємодіяти з гравцем лише за 2 години 24 хвилини.

20

СЛАЙД 21

Для повторного тесту було використано процесор Intel I9 11900 (11 покоління), який мав нову архітектуру для штучного інтелекту – Intel Deep Learning Boost.

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.142857	0.285714	0.752381	0.266667	-0.257143	-0.847619	0.333333
2	State1	0.876190	2.135180	2.122956	-0.847619	0.000000	0.790476	0.000000
3	State2	0.476191	0.323810	1.067830	0.000000	1.120466	0.400000	0.000000
4	State3	2.152290	-0.361905	0.961905	-0.742857	0.485714	-1.197333	-0.876190
5	State4	-1.242607	-0.838095	1.109596	0.047619	-1.528828	0.371429	1.861733
6	State5	1.456522	2.128169	-1.876217	1.327616	-4.974322	2.183065	5.696792
7	State6	1.089053	-1.009524	0.828571	0.000000	0.942857	0.000000	0.000000

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.209524	0.761905	0.466667	-0.114286	-0.247619	0.085714	0.647619
2	State1	0.152381	0.723810	0.428571	-0.028571	0.009524	-0.200000	-0.066667
3	State2	0.171428	0.047619	-0.047619	0.161905	-0.076190	0.380952	-0.104762
4	State3	0.114286	0.266667	0.342857	0.047619	0.152381	-0.095238	-0.019048
5	State4	0.228571	0.161905	-0.161905	0.095238	0.171429	0.047619	-0.114286
6	State5	0.000000	0.466667	0.104762	0.400000	-0.200000	0.285714	0.057143
7	State6	-0.152381	0.161905	0.238095	0.200000	0.152381	0.000000	0.000000

21

СЛАЙД 22

Для повторного тесту було використано процесор Intel I9 11900 (11 покоління), який мав нову архітектуру для штучного інтелекту – Intel Deep Learning Boost.

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.000000	0.495238	0.390476	0.333333	-0.314286	0.000000	0.000000
2	State1	0.000000	0.000000	0.228571	-0.028571	0.247619	-0.085714	0.047619
3	State2	0.000000	0.619048	0.438095	0.000000	0.238095	-0.438095	-0.180952
4	State3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	State4	0.200000	-0.114286	0.266667	-0.238095	0.285714	0.200000	0.133333
6	State5	1.131137	0.885714	0.085714	0.057143	0.400000	0.057143	0.000000
7	State6	0.000000	0.000000	0.000000	0.000000	0.514286	0.314286	0.000000

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.961905	-0.171429	0.314286	-0.495238	-0.276190	0.238095	0.209524
2	State1	0.266667	0.257143	0.219048	-0.028571	-0.190476	-0.085714	0.180952
3	State2	0.095238	0.238095	0.133333	0.200000	0.171429	-0.152381	-0.161905
4	State3	0.857143	-0.523810	0.466667	0.400000	-0.400000	0.609524	-0.295238
5	State4	0.323810	-0.257143	0.000000	0.342857	-0.514286	0.523810	0.495238
6	State5	-0.171429	0.104762	0.200000	0.152381	0.285714	0.257143	-0.295238
7	State6	-0.161905	-0.314286	0.133333	0.161905	0.209524	0.114286	0.238095

Повторне навчання з використанням цього апаратного рішення зайняло лише 1 годину і 7 хвилин, а звільнені ресурси можна було переделегувати на інші процеси.

22

СЛАЙД 23

Висновки

- У роботі за результатами виконаних теоретичних та практичних досліджень набула подальшого розвитку модель ШІ–прискорювача в сучасних процесорах для обрахунку штучного інтелекту в іграх, для обрахунку штучного інтелекту в іграх, що використовує процесор Intel I9 11900 (11 покоління), а також удосконаленний агентно–орієнтований метод, що дозволяє підвищити продуктивність агента за рахунок врахування вагового коефіцієнта винагороди, що в свою чергу задовільняє функціональну та дизайнерьску систему гри. На основі запропонованого підходу розроблена інтелектуалізована система з використанням процесора Intel I9 11900, що дозволяє при застосуванні методів машинного навчання для розроблення комп'ютерних ігор, підвищити ефективність та швидкість навчання персонажів ігрових проектів.

23

СЛАЙД 24

Висновки

- Проаналізувавши результати навчання штучного інтелекту можна зробити висновок, що використання процесору Intel I9 11900 (11 покоління), який мав нову архітектуру для штучного інтелекту – Intel Deep Learning Boost виявилося найбільш ефективним. Повторне навчання з використанням цього апаратного рішення зайняло лише 1 годину і 7 хвилин, а звільнені ресурси можна було переделегувати на інші процеси.
- Ефективність використання запропонованих рішень доведено експериментами. Використання моделі прискорювача штучного інтелекту дозволило в 2,14 рази пришвидшити навчання персонажу коп'ютерної гри порівняно з класичними методами.

24

СЛАЙД 25

Дякую за увагу!

25

ДОДАТОК В

Стаття

UDK 004.875; 004.97

Yelyzaveta Hnatchuk, Yevheniy Sierhieiev, Alina Hnatchuk

Khmelnytskyi National University, Khmelnytskyi, Ukraine

USING ARTIFICIAL INTELLIGENCE ACCELERATORS TO TRAIN COMPUTER GAME CHARACTERS

A review of the literature has shown that today, given the complexity of computational processes and the high cost of these processes, the gaming computer industry needs to improve hardware and software to increase the efficiency and speed of processing artificial intelligence algorithms. An analysis of existing machine learning tools and existing hardware solutions to accelerate artificial intelligence. A reasonable choice of hardware solutions that are most effective for the implementation of the task. Possibilities of practical use of the artificial intelligence accelerator are investigated. The effectiveness of the proposed solutions has been proven by experiments. The use of an artificial intelligence accelerator model allowed to accelerate the learning of a computer game character by 2.14 times compared to classical methods.

Keywords: processor, artificial intelligence, computer games, agent, machine learning.

Є.Г. Гнатчук, Є.В. Сєргєєв, А.Я. Гнатчук

Хмельницький національний університет

ВИКОРИСТАННЯ ПРИСКОРЮВАЧІВ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ НАВЧАННЯ ПЕРСОНАЖІВ КОМП'ЮТЕРНИХ ІГОР

В статті проаналізовано існуючі інструментальні засоби для полегшення та пришвидшення процесу розробки ігор. Однією з головних проблем, з якими стикаються розробники таких систем та додатків, є те, що алгоритми штучного інтелекту є обчислювально дорогими. Цей факт змушує шукати шляхи ефективного вирішення цієї проблеми, зокрема використання поліпшення апаратного прискорення, щоб забезпечити необхідну високу обчислювальну потужність. Оптимізована та спеціалізована апаратна реалізація може зменшити системні витрати за рахунок оптимізації необхідних ресурсів та зменшення вимог до енергії при одночасному поліпшенні продуктивності. Все більше дослідників, зокрема зарубіжних, досліджують галузь та пропонують нові інструментальні засоби для полегшення та пришвидшення процесу розробки ігор.

Враховуючи складність обчислювальних процесів та велику вартість цих процесів, ігрова комп'ютерна галузь потребує вдосконалення апаратного та програмного забезпечення для підвищення ефективності та швидкості обробки алгоритмів штучного інтелекту. Доведено, що використання прискорювачів штучного інтелекту для навчання персонажів комп'ютерних ігор з використанням методів машинного навчання є актуальною задачею. Це дозволяє підвищити ефективність та швидкість обробки алгоритмів штучного інтелекту. Проведений аналіз показав, що для конкретної задачі, що розглядається в даній роботі, доцільним для прискорення навчання ШІ є використання моделі прискорювача штучного інтелекту, що використовує процесор Intel I9 11900 (11 покоління), який має нову архітектуру для штучного інтелекту – Intel Deep Learning Boost. Досліджено можливості практичного використання прискорювача штучного інтелекту. Ефективність використання запропонованих рішень доведено експериментами. Використання моделі прискорювача штучного інтелекту дозволило в 2,14 рази пришвидшити навчання персонажу комп'ютерної гри порівняно з класичними методами.

Ключові слова: процесор, штучний інтелект, комп'ютерні ігри, агент, машинне навчання.

Introduction

Artificial intelligence (AI) and machine learning tools have become widespread in recent years, thanks to advances in computing in power, computing, and performance.

Artificial intelligence is used in an extremely wide range of programs to get better results compared to traditional methods.

Such applications include image processing, such as face detection and recognition [1], financial markets analysis and banking [2], robotic systems in industry [3], medical applications and healthcare applications [4], efficient transactions in database management [5], security applications [6], unmanned delivery services and personal transport [7], autonomous drones for navigation [8] and much more.

Given the requirements of such applications for accuracy and efficiency, many of these applications are based on artificial intelligence. This trend indicates the constant interest and high potential of artificial intelligence tools and machine learning.

These tools are increasingly becoming an integral part of every electronic or embedded system. Microrobots with built-in artificial intelligence are widely used in practice in various fields. The robots include advanced controllers that implement AI functions, such as perception (information retrieval) and cognition (decision making). One of the main problems faced by developers of such systems and applications is that artificial intelligence algorithms are computationally expensive. This fact leads to the search for ways to effectively solve this problem, in particular the use of improved hardware acceleration to provide the necessary high computing power.

Optimized and specialized hardware implementation can reduce system costs by optimizing the required resources and reducing energy requirements while improving productivity [9].

Related works

In [11] a study of the hardware implementation of artificial intelligence algorithms and machine learning from 2009 to 2019. The main purpose of this work was to introduce neural networks as a tool for detecting and recognizing objects in various applications. In this study, the results of two hundred and one scientific works are analyzed and presented, 169 works were related to the hardware implementation of algorithms of artificial intelligence and machine learning. Hardware acceleration is considered an ideal solution for energy-intensive and resource-intensive artificial intelligence algorithms.

The aim is to achieve faster and more efficient processing of AI algorithms [12]. In recent years, many studies have been published that discuss a large number of implementations of hardware and software optimization and implementation methods in this area [13-20]. Part of the research looked at the implementation of artificial neural networks in hardware in general, other studies focused on FPGA accelerators for deep learning neural networks. For example, in [15] the authors focused on the implementation of FPGA wrapped neural networks. In [17], the study discussed the details of the implementation of the GPU.

In [21] all the main hardware implementations of artificial neural networks implemented on hardware, called hardware neural networks (ASM), were studied. Hardware neural networks are classified according to some characteristics, such as embedded / disabled chip, analog / digital blocks, thresholds, lookup table, calculation and data rate. A special section for ASM chips covers the implementation of digital and hybrid neurochips based on FPGA and ASIC.

In [22] a more detailed study was conducted, the main aspects of which were devoted to neural network accelerators based on FPGA, where they investigated the design of neural network accelerators and summarized all the methods used to automate the design of accelerators.

The study [24] discussed the basics of learning deep neural networks with this section for methods of implementing compression. The survey analyzes FPGA-based accelerators and ASIC-based accelerators, with a greater emphasis on FPGAs. Other studies, such as [25], have also considered techniques for designing FPGA-based accelerators for compression neural networks.

In conclusion, we can say that all studies have examined various alternatives to the GPU in a comprehensive way to improve the performance of artificial intelligence algorithms. The use of GPUs is a good option in terms of accelerating artificial intelligence and an option that can compete with solutions based on FGPA and ASIC.

A review of the literature has shown that the popularity of computer games is growing today. More and more researchers, including foreign ones, are researching the industry and offering new tools to facilitate and speed up the game development process.

Given the complexity of computational processes and the high cost of these processes, the gaming computer industry needs to improve hardware and software to increase the efficiency and speed of processing artificial intelligence algorithms.

Therefore, the use of artificial intelligence accelerators to train computer game characters based on machine learning techniques is an urgent task.

Using microcomputers for machine learning

An important issue is the choice of technology that increases the productivity and efficiency of artificial intelligence in games. Interesting for practical use is Intel's technology designed to automatically increase the clock speed of the processor above the nominal, which is called Turbo Boost. However, the power, temperature and current limits in the design capacity must not be exceeded. The use of such technology increases the performance of single-stream and multi-stream programs.

In other words, it is actually a processor overclocking technology. The availability of Turbo Boost technology does not depend on the number of active cores. It depends on the presence of one or more cores that can operate at a power that is below the design. The operating time of the system in this technology depends on such things as workload, platform design and operating conditions. Intel Turbo Boost technology is enabled in one of the BIOS menus by default.

Another technology is a neural processor or artificial intelligence accelerator (NPU). This is a specialized class of microprocessors or otherwise coprocessors, which is most often used to hardware accelerate the work of artificial intelligence. Such as artificial neural network algorithms, voice recognition, computer vision, machine learning and other artificial intelligence methods. Neural processors are computer technology and are used to hardware accelerate the emulation of neural networks, as well as real-time digital signal processing.

Most often, neural processors include store-type memory blocks, registers, a computing device containing a multiplication matrix, a switch, decoders, multiplexers, and triggers.

The class of neural processors includes:

- Neuromorphic processors built on a cluster architecture. Unlike traditional computing architectures, they are highly specialized for the creation and development of various types of artificial neural networks. These processors use conventional transistors. Computational cores are built from them. Each kernel contains a router to communicate with other kernels, a task scheduler, and its own SRAM. Each nucleus emulates the work of a large number of neurons. Such processors are used for deep machine learning.
- Tensor processors, which are coprocessors, are controlled by a CPU operating with tensors. They have their own built-in RAM and are highly specialized for performing matrix multiplication and convolution operations. These operations are used to emulate wrapped neural networks used for machine learning.
- Machine vision processors have many similarities with tensor processors. But they differ from them in that they are used to accelerate the operation of machine vision algorithms, which use the methods of convolutional networks, as well as scale-invariant transformation of features. The emphasis is on parallelizing the flow of data between multiple executive cores. They, like tensors, are used for low-precision calculations, which is accepted in image processing.

In the case of machine learning using microcomputers, note that the reinforced learning model cannot learn on a RaspberryPi or using another microcomputer because their boards do not have enough power to perform a large number of operations. During training, a large number of floating-point multiplication operations take up all the computing resources of the microcomputer, so it is only possible to import and run a ready-made model on microcomputers.

Let's consider the basic boards of microcomputers, their parameters and possibilities of additional modules for machine learning. In [15] the review of comparison of efficiency of microcomputer platforms of machine learning is given. The review concluded that the most affordable solution for learning at an affordable price would be a combination of a Raspberry Pi 4 microcomputer with an Intel Neural Compute Stick 2 and a TensorFlow framework.

TensorFlow from Google is the open and most popular deep learning library for research and development. TensorFlow is a cross-cutting platform that simplifies the creation and deployment of ML models. TensorFlow offers several levels of abstraction so you can choose the right one for your needs. You can create and train models using the high-level Keras API, which makes it easy to get started with TensorFlow and machine learning. If more flexibility is required, persistent execution allows immediate iteration and intuitive debugging.

For large machine learning tasks, distribution strategy APIs are used for distributed learning on different hardware configurations without changing the model definition.

An important issue in the calculation of artificial intelligence in games and the use of machine learning is acceleration, which allows you to effectively address the allocation of CPU resources and reduce the learning time of artificial intelligence.

Consider the existing and most commonly used hardware accelerators of artificial intelligence and machine learning, based on standard quality indicators. Hardware accelerator documents are classified from a hardware perspective to provide a useful comparison. These are the five categories listed below and shown in Figure 1:

- hardware mapping;
- accelerator design based on FPGA;
- accelerator design based on GPU;
- reclassification of accelerators;
- Xeon-Phi accelerator design.

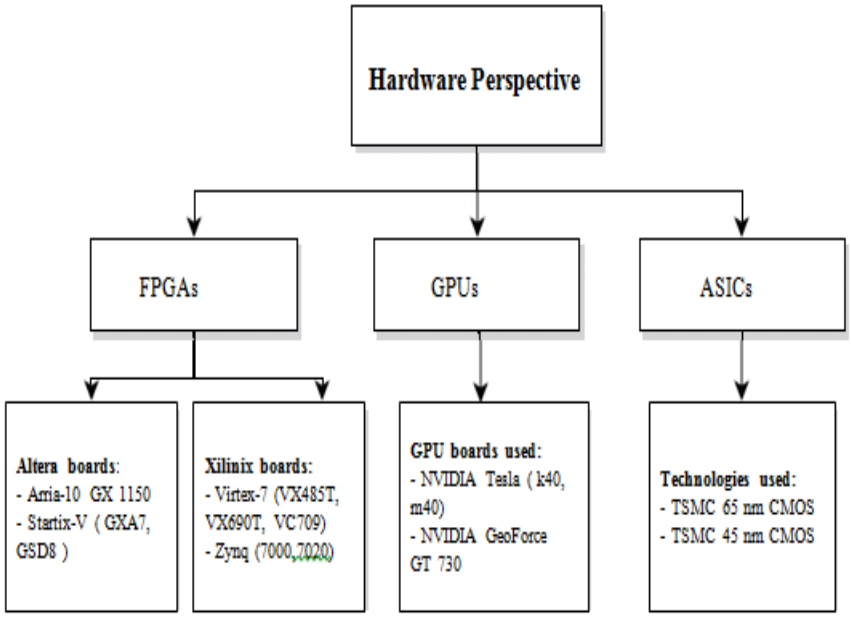


Fig. 1 - Hardware accelerators of artificial intelligence

Consider in more detail the scope of each of them. The problem facing researchers is to find an appropriate algorithm for a particular application and its effective application to the hardware. CNNs are built using direct transmission networks, which can be considered as a large number of similar scalar operations transmitted at different stages. This type of calculation is significantly accelerated when working on FPGA. FPGA mapping tasks include finding an effective match between the CNN computing model and the execution model supported by the FPGA. Several research papers have addressed these issues of CNN mapping on FPGA.

The heuristic search algorithm and data stream command set architecture used DNNWEAVER to create high-performance accelerators running on a limited energy budget and built-in FPGA memory. DNNWEAVER is a structure that automatically generates a synthesized accelerator for a given pair (DNN, FPGA) using manually optimized templates.

The Deep Burning structure is built to simplify the operation of displaying various neural networks in FPGA or ASIC. RTL library compiler for automatic creation of individual FPGA-based accelerators for this AI algorithm.

The possibility of direct hardware mapping of AI on FPGA is considered by various researchers. The proposed model states that it opens up new opportunities for further optimization and can be extended to ASIC technology, as well as binary neural networks. There is a delay-based design methodology for mapping ConvNets to FPGAs. Python-based automation tools for creating FPGA-based AI accelerators are quite popular to use. The tool constantly provides high throughput for various AI models.

The RTL library compiler achieved the best performance: 732.36 GOPS, 1604.57 GOPS, 651.49 GOPS and 789.44 for NiN, VGG-16, ResNet-50 and ResNet-152 on Startix 10 GX2800 FPGA respectively. For AlexNet, the python-based automation tool showed the best performance, which is 274.5 GOPS.

Many FPGA-based accelerator architecture designs have been proposed. Most of the research work was based on FNGA-based CNN accelerators. From a hardware point of view, these documents are divided into two categories:

- heterogeneous architecture;
- FPGA.

Heterogeneous architecture accelerators use a high-quality FPGA processor to implement a high-performance binary neural accelerator (BNN). The proposed accelerator is designed to take advantage of the Xeon CPU + FPGA system. The FPGA architecture is designed for the most computational parts of the BNN, while other parts of the topology can be processed by the CPU. There are other AI accelerators based on CPU-FPGA. Intensive computational and universal operations (eg, 2D convolutions) were implemented in the FPGA.

Other heterogeneous architectures combining ASIC with FPGA have focused on deep learning with efficient tensor matrix / vector operations. The application of the accelerator architecture using two FPGAs is promising. Implementation of the accelerator based on AI FPGA reached an accuracy of 82.8%.

Xilinix FPGA-based AI accelerators have the highest performance among other implementations. Their implementation provides the latest results. This is due to the use of interlayer planning, which has reduced data transfer from 6.6 MB to 0.2 MB with a reduction of 97%.

FPGA-based AI accelerators show the highest performance. This architecture achieves high performance due to several sources of parallelism integrated into the implementation of AI, which aim to achieve the best possible acceleration.

Other FPGA-based AI accelerators have been implemented using altera boards. As for the implementation with a fixed point, it is known about the highest efficiency among other implementations. They addressed the performance constraints caused by the low bandwidth of the built-in memory due to the two-dimensional relationship between the processor elements (CPUs) and the local memory. This effectively increases the effective bandwidth of the built-in memory. Whereas for 32 floating-point implementations on altera boards, the highest performance is reported.

The methods used to achieve this high performance were to use a built-in flow buffer that efficiently stores input and output function maps. In addition, a vectorization approach was used, which achieves more than 60% DSP efficiency.

Built-in FPGA processors, such as NIOS-II, and ARC processors in SoC-based FPGAs are also used to implement the target accelerators. AI acceleration can also be achieved by using parallel programmable logic to implement convolution, merging, and add-ons, as well as a built-in ARM processor to run and perform other tasks.

GPUs are becoming important to accelerate deep learning. Existing methods are aimed at accelerating algorithms on GPUs.

Over the past decade, Intel Xeon Phi coprocessor has paved the way for success in implementing deep learning algorithms. Intel Xeon Phi is a shared memory. This means a multi-core coprocessor that contains up to 61, 1.2 GHz cores, and each core can switch between 4 hardware threads in a circular manner. Xeon Phi makes extensive use of Single Instruction Multiple Data (SIMD) technology, which allows you to perform the same operation on multiple data segments simultaneously. Thus, Xeon Phi is very suitable for deep learning programs, which usually use simple operations on large tensors.

Researchers have shown that Intel Xeon Phi can offer an efficient but more general way to parallelize the deep learning algorithm compared to GPUs. In-depth learning can be effectively optimized and scaled on multi-core HPC systems.

The proposed CosmowFlow, the first large-scale scientific application of the TensorFlow framework on a supercomputer scale, is completely synchronous. Optimized full software stack that includes network design, I / O processing, communication, TensorFlow framework, and 3D CNN placement in MKLDNN for 3D convolutional neural network. Some studies by mid-level researchers on BNN have proposed a new approach to optimizing BNN on processors using the BitFlow framework BitFlow gets 1.8 accelerations.

Hardware technologies for machine learning:

- programmable FP-DNN, a structure having the input data of the described TensorFlow DNN, used to generate hardware implementations on FPGA boards with hybrid RTL-HLS templates;
- structure used to automate the display of AI on systolic arrays on FPGA;
- TuRF AI acceleration structure, inspired by efficient AI architectures and transfer training, which supports optimization for a specific domain. It efficiently uses domain programs on the FPGA;
- Caffe uses an open source deep learning system to provide clear access to deep architectures. The code is written in pure, efficient C ++, where CUDA is used to calculate the GPU;
- clCaffe, an acceleration of OpenCL's well-known Caffe deep learning mechanism, while focusing on the convolution level, which has been optimized by three different approaches. This greatly improves the ability to use deep learning cases on all types of OpenCL devices.

There are many open source tools supported by NVIDIA and Intel to support accelerating machine learning algorithms. For example, NIVIDIA has developed RAPIDS, a machine learning library with an accelerated graphics processor, which aims to significantly speed up the learning process. RAPIDS improves performance by accelerating the complete pipeline pipeline, including data preparation, machine learning algorithm, and GPU visualization.

RAPIDS also significantly speeds up processing, trains larger datasets, and supports the deployment of multiple GPUs. Fast is used to successfully reduce the required training time of systems that recommend what is computationally considered intensive with an acceleration of 15.6 times.

The analysis showed that for the specific task considered in this paper, it is advisable to accelerate the learning of AI is to use a model of artificial intelligence accelerator using Intel I9 11900 processor (11th generation), which had a new architecture for artificial intelligence - Intel Deep Learning Boost .

Experiments

To prove the effectiveness of the use of artificial intelligence accelerators in computer games, an experiment was conducted to train a shock-blocking agent. This paper discusses the following key mechanics of a computer game. These are fist fights. If you omit the means of movement and orientation, the player will have access to 8 actions and 8 pathogens from which the decision will be made on the following moves:

9. Nothing.
10. Left Attack.
11. Right Attack.
12. Left Block.
13. Right Block.
14. Left Fint.
15. Right Fint.
16. Grab.

	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint	Grab
Nothing	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Left Attack	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Right Attack	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Left Block	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Right Block	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Left Fint	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Right Fint	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Grab	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig.2 - Initial table for the agent

Artificial intelligence changes the state in the following two cases:

1) when the player performed the action and the artificial intelligence was notified;

2) every 2 seconds.

Artificial intelligence checks the received condition (for example, the player has executed blow with the left hand), addresses to the table (Fig. 2) and chooses optimum action. Then apply the selected action, after its completion receives the result and updates the table.

In this paper, the testing of artificial intelligence was performed in several ways: manually, using a simple calculation by the processor, as well as using a processor to accelerate artificial intelligence.

An example of a table with the results of manual training is shown in Figure 3.

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	1.242522	0.992239	-3.471755	-0.971019	-0.514808	0.000000	0.000000
2	State1	0.000000	1.000000	0.000000	0.000000	2.603926	0.000000	0.000000
3	State2	-0.250000	0.000000	2.000000	3.274837	0.000000	0.000000	0.000000
4	State3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	State4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
6	State5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7	State6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Fig. 3 – Table with the results of manual training

Subsequent testing of artificial intelligence using reinforcement training was performed by processing the standard method using dry processor computing.

As a result, the agent learned to interact with the player in just 2 hours and 24 minutes.

The Intel I9 11900 (11th generation) processor, which had a new architecture for artificial intelligence – Intel Deep Learning Boost – was used for re-testing.

From the presented table, artificial intelligence chose to attack with the left hand. Due to the fact that the player rarely makes a block with the right hand, the character of the computer game is injured. As for blocking, he learned to block the player's shots after 15 shots.

The results of reinforcement training using a processor to accelerate artificial intelligence are shown in Figures 4 - 7.

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.142857	0.285714	0.752381	0.266667	-0.257143	-0.847619	0.333333
2	State1	0.876190	2.135180	2.122956	-0.847619	0.000000	0.790476	0.000000
3	State2	0.476191	0.323810	1.067830	0.000000	1.120466	0.400000	0.000000
4	State3	2.152290	-0.361905	0.961905	-0.742857	0.485714	-1.197333	-0.876190
5	State4	-1.242607	-0.838095	1.109596	0.047619	-1.528828	0.371429	1.861733
6	State5	1.456522	2.128169	-1.876217	1.327616	-4.974322	2.183065	5.696792
7	State6	1.089053	-1.009524	0.828571	0.000000	0.942857	0.000000	0.000000

Fig. 4 – The results of experiments

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.209524	0.761905	0.466667	-0.114286	-0.247619	0.085714	0.647619
2	State1	0.152381	0.723810	0.428571	-0.028571	0.009524	-0.200000	-0.066667
3	State2	0.171428	0.047619	-0.047619	0.161905	-0.076190	0.380952	-0.104762
4	State3	0.114286	0.266667	0.342857	0.047619	0.152381	-0.095238	-0.019048
5	State4	0.228571	0.161905	-0.161905	0.095238	0.171429	0.047619	-0.114286
6	State5	0.000000	0.466667	0.104762	0.400000	-0.200000	0.285714	0.057143
7	State6	-0.152381	0.161905	0.238095	0.200000	0.152381	0.000000	0.000000

Fig. 5 – The results of experiments

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.000000	0.495238	0.390476	0.333333	-0.314286	0.000000	0.000000
2	State1	0.000000	0.000000	0.228571	-0.028571	0.247619	-0.085714	0.047619
3	State2	0.000000	0.619048	0.438095	0.000000	0.238095	-0.438095	-0.180952
4	State3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	State4	0.200000	-0.114286	0.266667	-0.238095	0.285714	0.200000	0.133333
6	State5	1.131137	0.885714	0.085714	0.057143	0.400000	0.057143	0.000000
7	State6	0.000000	0.000000	0.000000	0.000000	0.514286	0.314286	0.000000

Fig. 6 – The results of experiments

	Row Na	Nothing	Left Attack	Right Attack	Left Block	Right Block	Left Fint	Right Fint
1	State0	0.961905	-0.171429	0.314286	-0.495238	-0.276190	0.238095	0.209524
2	State1	0.266667	0.257143	0.219048	-0.028571	-0.190476	-0.085714	0.180952
3	State2	0.095238	0.238095	0.133333	0.200000	0.171429	-0.152381	-0.161905
4	State3	0.857143	-0.523810	0.466667	0.400000	-0.400000	0.609524	-0.295238
5	State4	0.323810	-0.257143	0.000000	0.342857	-0.514286	0.523810	0.495238
6	State5	-0.171429	0.104762	0.200000	0.152381	0.285714	0.257143	-0.295238
7	State6	-0.161905	-0.314286	0.133333	0.161905	0.209524	0.114286	0.238095

Fig. 7 – The results of experiments

After analyzing the results of artificial intelligence training using all three methods, we can conclude that the use of Intel I9 11900 processor (11th generation), which had a new architecture for artificial intelligence - Intel Deep Learning Boost was the most effective.

Retraining using this hardware solution took only 1 hour and 7 minutes, and the released resources could be reallocated to other processes.

The effectiveness of the proposed solutions has been proven by experiments. The use of the model of the artificial intelligence accelerator allowed to accelerate the learning of the character of the computer game by 2.14 times compared to the classical methods.

Conclusions

A review of the literature has shown that today, given the complexity of computational processes and the high cost of these processes, the gaming computer industry needs to improve hardware and software to increase the efficiency and speed of processing artificial intelligence algorithms. An analysis of existing machine learning tools and existing hardware solutions to accelerate artificial intelligence. A reasonable choice of hardware solutions that are most effective for the implementation of the task. Possibilities of practical use of the artificial intelligence accelerator are investigated.

The effectiveness of the proposed solutions has been proven by experiments. The use of an artificial intelligence accelerator model allowed to accelerate the learning of a computer game character by 2.14 times compared to classical methods.

References

54. Sze V, Chen Y-H, Yang T-J, Emer J S. Efficient processing of deep neural networks: a tutorial and survey. *Proc IEEE*, 2017. Vol. 105(12). Pp. 2295–2329.

55. Yao X, Zhou J, Zhang J, Boer C R. From intelligent manufacturing to smart manufacturing for industry 4.0 driven by next generation artificial intelligence and further on. In: *5th International Conference on Enterprise Systems (ES)*, 2017. Pp. 311 – 318.

56. Bishnoi L, Narayan Singh S. Artificial intelligence techniques used in medical sciences: a review. In: *8th International Conference on Cloud Computing, Data Science and Engineering (Confluence)*, 2018. Pp. 106–113.

57. Rao Q, Frtunikj J. Deep learning for self-driving cars. In: *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems—SEFAIS '18*, 2018. Pp. 35–38. doi:10.1145/3194085.3194087.

58. Baji T. Evolution of the GPU device widely used in AI and massive parallel processing. In: *IEEE 2nd Electron Devices Technology and Manufacturing Conference (EDTM)*, 2018. Pp.7–9. doi:10.1109/EDTM.2018.8421507.

59. Shawahna A, Sait SM, El-Maleh A. FPGA-based accelerators of deep learning networks for learning and classification: a review. *IEEE Access*, 2019. Vol. 7. Pp. 7823–7859.

60. Mittal S. A survey of FPGA-based accelerators for convolutional neural networks. *Neural Comput Appl*, 2018. Vol. 32(4). Pp.1109–1139.

61. Apple : (документація) / Classifying Images with Vision and Core ML – 2020. URL: https://developer.apple.com/documentation/vision/classifying_images_with_vision_and_core_ml (дата звернення: 11.02.2021).

62. Jawandhiya P. Hardware design for machine learning. *Int J Artif Intell Appl (IJAIA)*, 2018. Vol. 9(1). Pp. 63–84.

63. Wang T, Wang C, Zhou X, Chen H. A survey of FPGA based deep learning accelerators: challenges and opportunities, *CoRR*, 2018. Vol. 1901.04988.

64. Rigos S. A hardware acceleration unit for face detection. In: *Mediterranean Conference on Embedded Computing (MECO)*, Bar, 2012. Pp. 17–21.

65. Nurvitadhi E, Venkatesh G, Sim J, Marr D, Huang R, Ong Gee Hock J, Liew YT, Srivatsan K, Moss D, Subhaschandra S, Boudoukh G. Can FPGAs beat GPUs in accelerating next-gen- eration deep neural networks? In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays—FPGA '17*, 2017. Pp. 5–14. doi:10.1145/3020078.3021740.

66. Lacey G, Taylor G, Areibi S. Deep learning on FPGAs: past, present, and future, *CoRR*, 2016. Pp. 1–8. arXiv: 1602.04283.

67. Faraone J, Gambardella G, Boland D, Fraser N, Blott M, Leong PHW. Customizing low-precision deep neural networks for FPGAs. In: *28th International Conference on Field Programmable Logic and Applications (FPL)*,

IEEE, 2018. Pp. 97–102.

68. Mogilny S.B. Machine learning using microcomputers: teaching method. way. / for ed. O.V. Lisovogo and others.

69. Cheng Kwang-Ting, Wang Yi-Chu. Using mobile GPU for general-purpose computing; a case study of face recognition on smartphones. In: *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test*, 2011. Pp. 1–4, doi: 10.1109/VDAT.2011.5783575.

70. Ouerhani Y, Jridi M, Al Falou A. Fast face recognition approach using a graphical processing unit "GPU". In: *IEEE International Conference on Imaging Systems and Techniques*, 2010. Pp.80–84.

71. Li E, Wang B, Yang L, Peng Y, Du Y, Zhang Y, Chiu Y-J. GPU and CPU cooperative accel-eration for face detection on modern processors. *Presented at the 2012 IEEE International Confer- ence on Multimedia and Expo (ICME)*, 2012. Pp. 769–775.

72. Shah A. A, Zaidi Z. A, Chowdhry B. S, Daudpoto J. Real time face detection/monitor using raspberry pi and MATLAB. In: *IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, 2016. Pp. 1–4.

73. Oro D, Fernandez C, Saeta J. R, Martorell X, Hernando J. Real-time GPU-based face detec- tion in HD video sequences. In: *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011. Pp. 530–537.

74. Misra J, Saha I. Artificial neural networks in hardware: a survey of two decades of progress. *Neurocomputing*, 2010. Vol. 74(1–3). Pp.239–255.

75. Guo K, Zeng S, Yu J, Wang Y, Yang H. A survey of FPGA-based neural network inference accelerators. *ACM Trans Reconfig Technol Syst*, 2019. Vol. 12(1). Pp. 1–26.

76. Talib, M.A., Majzoub, S., Nasir, Q. et al. A systematic literature review on hardware implementation of artificial intelligence algorithms. *J Supercomput*, 2021. Vol. 77. Pp. 1897–1938. doi:10.1007/s11227-020-03325-8.

Yelyzaveta Hnatchuk – PhD, Associate Professor of Computer Engineering & System Programming Department, Khmelnytskyi National University, Khmelnytskyi, Ukraine,

e-mail: liza_veta@ukr.net.

orcid.org/ 0000-0003-2989-3183,

Scopus Author ID: 57211621395,

ResearcherID: I-1504-2018,

https://scholar.google.com.ua/citations?hl=ru&user=5tG01jkAAAAJ.

Yevheniy Sierhieiev – master of Computer Engineering, Computer Engineering & System Programming Department, Khmelnytskyi National University, Khmelnytskyi, Ukraine,

e-mail: ysierhieiev@gmail.com

Alina Hnatchuk – student of Computer Engineering, Computer Engineering & System Programming Department, Khmelnytskyi National University, Khmelnytskyi, Ukraine,

e-mail: alinasamsungj5gold2001@gmail.com,

orcid.org/0000-0003-0155-9255.

Гнатчук Єлизавета Геннадіївна – кандидат технічних наук, доцент кафедри комп'ютерної інженерії та системного програмування, Хмельницький національний університет, Хмельницький, Україна.

Сєргєєв Євгеній Віталійович – магістр спеціальності комп'ютерна інженерія, Хмельницький національний університет, Хмельницький, Україна.

Гнатчук Аліна Ярославівна – студентка спеціальності комп'ютерна інженерія, Хмельницький національний університет, Хмельницький, Україна.

Довідка

Видана Сєргєєву Є.В., що стаття «Using artificial intelligence accelerators to train computer game characters» авторів Гнатчук Є.Г., Сєргєєв Є.В., Гнатчук А.Я, співавтором якої він є, прийнято та буде опубліковано в журналі Computer Systems and Information Technologies №1 за 2021 р.

Головний редактор журналу Говорущенко Т.О.

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1007957962

Дата перевірки:
21.05.2021 09:36:11 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
21.05.2021 09:36:40 EEST

ID користувача:
100005591

Назва документа: Інтелектуалізована система на основі методів машинного навчання для розроблення комп...

Кількість сторінок: 86 Кількість слів: 14424 Кількість символів: 111752 Розмір файлу: 2.24 MB ID файлу: 1008050888

1.92%
Схожість

Найбільша схожість: 0.64% з джерелом з Бібліотеки (ID файлу: 1007657363)

1.28% Джерела з Інтернету	81	Сторінка 88
0.85% Джерела з Бібліотеки	62	Сторінка 88

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2

Fri May 21 08:49:46 EEST 2021, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХН

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 1.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 11%**

ID: 91103 Название: Інтелектуалізована система на основі методів машинного навчання для розроблення комп'ютерних ігор Добавлено в БД: 2021-05-21 Авторы: Сергєєв Є.В._ Руководители: Гнатчук Є.Г. Консультанты: Оппоненты:	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
	105411	820	987 (1%)	10 (1%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник: Сєргєєв Євгеній Віталійович

Тема: Інтелектуалізована система на основі методів машинного навчання для розроблення комп'ютерних ігор

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг дипломної роботи:

Кількість листів креслень – ; кількість сторінок записки 91

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано інтелектуалізовану систему на основі методів машинного навчання для розроблення комп'ютерних ігор

2. Висновок про відповідність роботи дипломному завданню _____ Дипломна робота відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі роботи був проведений аналіз існуючих інструментальних засобів для полегшення та пришвидшення процесу розробки ігор. Доведено, що використання прискорювачів штучного інтелекту для навчання персонажів комп'ютерних ігор за допомогою інтелектуалізованої системи на основі методів машинного навчання дозволяє підвищити ефективність та швидкість обробки алгоритмів штучного інтелекту. У другому розділі роботи представлена формалізація процесу прийняття рішення з використанням ланцюгів Маркова та модель агента, а також представлений удосконаленний агентно-орієнтований метод, що дозволяє підвищити продуктивність агента за рахунок врахування вагового коефіцієнта винагороди, що в свою чергу задовільняє функціональну та дизайнерську систему гри. У третьому розділі роботи, відповідно до розробленої формалізованої моделі агента та агентно-орієнтованого методу, виконано проектування структури ПЗ та розроблено алгоритм роботи інтелектуалізованої системи. Проведений аналіз показав, що для конкретної задачі, що розглядається в даній роботі, доцільним для прискорення навчання ШІ є використання моделі прискорювача штучного інтелекту, що використовує процесор Intel I9 11900 (11 покоління), який мав нову архітектуру для штучного інтелекту – Intel Deep Learning Boost. В четвертому розділі здійснено обгрунтований вибір засобів розроблення ПЗ та апаратних рішень, які є найбільш ефективними для реалізації інтелектуалізованої системи на основі методів машинного навчання для розроблення комп'ютерних ігор.

4. Позитивні сторони роботи: Запропоновані результати використання моделі прискорювача штучного інтелекту дозволило в 2,14 рази пришвидшити навчання персонажу коп'ютерної гри порівняно з класичними методами. Отримані результати

магістерської роботи можуть бути використані для підвищення ефективності використання машинного навчання при розробці комп'ютерних ігор.

5. Негативні сторони роботи: В роботі не в повній мірі реалізовано експерименти на різних апаратних платформах.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Матеріали кваліфікаційної роботи є структурованими у чіткій та логічній формі та відображають послідовність виконання поставлених задач.

7. Відгук про роботу в цілому: Загалом, зміст представленої роботи в повній мірі розкриває обрану тему. Виконані дослідження, що представлені в поданій роботі є достатньо аргументованими. Теоретичні викладки підтверджені проведеними експериментами. Результатом проведення досліджень стали відповідні висновки і конкретні пропозиції щодо удосконалення агентно-орієнтованого методу та модель прискорювача штучного інтелекту, що дозволили підвищити ефективність

8. Інші зауваження: –

9. Оцінка дипломної роботи:

Розглянувши представлену дипломну роботу вважаю, що робота заслуговує оцінки «відмінно» 5,0 (A), а її автор – присвоєння кваліфікації «магістра» з комп'ютерної інженерії.

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Кльоц Юрій Павлович, к.т.н., доцент, завідувач кафедрою кібербезпеки та комп'ютерних систем і мереж ХНУ

"_20_" ___травня_____2021р. _____

Завідувачу кафедри КІСП
д-р.техн.наук, проф. Говорущенко Т. О.

Сєргєєв Є.В.
ПІБ здобувача вищої освіти

ФПКТС, 2 курсу, групи КІ2м-19-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

20.05.2021 _____

дата підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Інтелектуалізована система на основі методів машинного навчання для розроблення комп'ютерних ігор

Автор: Сєргєєв Євгеній Віталійович

Спеціальність: 123 – Компютерна інженерія та програмування

Освітня програма: освітньо-наукова

Науковий керівник: Гнатчук Єлизавета Геннадіївна, к.т.н., доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 1.92% і з них 1.28% адресується до 81 першоджерела, а 0.85%, до 62 першоджерела що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Є. Г. Гнатчук

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КІСП _____ Т. О. Говорущенко