Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти *другий (магістерський)*

Дослідження підходів навчання з підкріпленням нейронних мереж у контрольованих імітаційних середовищах
(тема)

Виконав:
студент *2* курсу, групи *СПРм-22-1*
Ананьєв М. А.
(прізвище, ініціали)

Спеціальність *122 Комп'ютерні науки*
(код і повна назва спеціальності)

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва освітньої програми)

Керівник *проф. Гребеннік І. В.*
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ *проф. Гребеннік І. В.*
(підпис) (прізвище, ініціали)

2024 р.

Я, як студент ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Ананьєв М. А.

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведено _____

Керівник кваліфікаційної роботи проф. Гребеннік І. В.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*

Кафедра *Системотехніки*

Рівень вищої освіти *другий (магістерський)*

Спеціальність *122 Комп'ютерні науки*
(код і повна назва)

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20___

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Ананьєву Максиму Анатолійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження підходів навчання з підкріпленням нейронних мереж у контрольованих імітаційних середовищах*

затверджена наказом університету від 18 січня 2024 р. № 44 Ст

2. Термін подання студентом роботи до екзаменаційної комісії ____ _____ 20__ р.

3. Вихідні дані до роботи Дослідити методи навчання з підкріпленням для нейронних мереж у контрольованих імітаційних середовищах. Розробити декілька імітаційних сцен для тренування агентів. Провести порівняння якості навчання агентів у різних середовищах. Розробити програмне рішення для автоматизації процесу навчання та порівняння агентів.

4. Перелік питань, що потрібно опрацювати в роботі Теоретичні та методологічні основи дослідження; Огляд існуючих підходів до навчання з підкріпленням; Вибір середовищ для імітаційного навчання; Розробка та налаштування імітаційних сцен; Проведення навчання агентів у вибраних середовищах; Порівняльний аналіз якості навчання агентів; Вивчення впливу параметрів середовища на процес навчання; Розробка критеріїв для оцінки ефективності навчання; Аналіз результатів експериментів; Розробка рекомендацій для покращення навчання агентів; Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 1.1. Схематичне представлення нейромережі. 1.2. Самокерований автомобіль. 1.3. Роботи доставки. 1.4. Серія моделей від команди GPT. 1.5. Приклад промислових роботів. 1.6. Персональні асистенти. 1.7. Приклад реклами на основі історії переглядів. 2.1. Схематичне зображення нейронної мережі. 2.2. Робота навчання з підкріпленням. 3.1. Спрощена блок-схема ML-Agents. 3.2. Декомпозиція навчального середовища. 3.3. Взаємодія агент-середовище в навчанні з підкріпленням. 3.4. Менеджер пакетів Unity. 3.5. Базовий шар. 3.6. Межі ігрового поля. 3.7. Приклад воріт однієї з команд. 3.8. М'яч. 3.9. Гравці різних команд. 3.10. Табло. 3.11. Приклад тестового середовища. 3.12. Налаштування вихідних параметрів. 3.13. Сенсори агента. 3.14. Перетин сенсорів із об'єктами. 3.15. Приклад налаштування сенсорів. 3.16. Доступна вхідна інформація агентів. 3.17. Інформація про стан навчання. 4.1. Встановлена модель. 4.2. Фінальний вигляд запущеної сцени. 4.3. Десять тестових середовищ. 4.4. Сцена один на один. 4.5. Сцена три на три гравці. 4.6. Сцена чотири на чотири гравці. 4.7. Сцена два на два гравці з вигнутим полем. 4.8. Поле з додатковими бар'єрами.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання завдання на виконання роботи*	*13.01.2024*	*Виконано*
2	*Вивчення літератури предметної області*	*15-23.01.2024*	*Виконано*
3	*Аналіз предметної області*	*24.01-02.02.2024*	*Виконано*
4	*Аналіз підходів вирішення проблеми*	*03.02.2024-28.02.2024*	*Виконано*
5	*Вибір мови та середовища розробки*	*01.03.2024*	*Виконано*
6	*Проектування та розробка програмного засобу*	*02.03-27.03.2024*	*Виконано*
7	*Проведення навчання*	*30.03-12.05.2024*	*Виконано*
8	*Обробка результатів навчання*	*13-18.05.2024*	*Виконано*
9	*Оформлення пояснювальної записки*	*19-26.05.2024*	*Виконано*
10	*Оформлення додатків*	*26.05.2024*	*Виконано*
11	*Представлення на рецензування*		

Дата видачі завдання *13 січня* 2024 р.

Студент _____
(підпис)

Керівник роботи _____ *проф. Гребеннік І. В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра містить: 85 сторінки, 34 рисунка, 29 таблиць, 26 джерел.

ІМІТАЦІЙНЕ СЕРЕДОВИЩЕ, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ

Об'єктом дослідження є процес навчання з підкріпленням нейронних мереж у контрольованих імітаційних середовищах, зокрема в середовищі Unity.

Предметом дослідження є методики машинного навчання з підкріпленням, використання технології Self-Play для навчання у віртуальному середовищі.

Метою дослідження є розробка та дослідження ефективності підходів проектування імітаційних сцен у Unity для тренування і порівняння нейронних мереж за допомогою Self-Play у контексті навчання з підкріпленням.

Методи дослідження – методи імітаційного моделювання, експерементальний підхід, методи статистичного аналізу

У роботі проведено аналіз сучасних підходів до навчання нейронних мереж у імітаційних середовищах, зокрема використання режиму Self-Play для автоматизації навчання. Також були розглянуті основні проблеми та виклики, пов'язані з навчанням з підкріпленням, та запропоновані шляхи їх вирішення.

Галузь застосування – ігрова індустрія, прикладні області з необхідністю автономного прийняття рішень, вдосконалення автономних транспортних систем.

ABSTRACT

Explanatory note to the qualification work of the master contains: 85 pages, 34 figures, 29 tables, 26 sources.

SIMULATION, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING

The object of research is the process of learning with reinforcement of neural networks in controlled simulation environments, in particular in the Unity environment.

The subject of the research is machine learning techniques with reinforcement and the use of Self-Play technology for learning in a virtual environment.

The purpose of the study is to develop and investigate the effectiveness of simulation scene design approaches in Unity for training and comparing neural networks using Self-Play in the context of reinforcement learning.

Research methods – methods of simulation modeling, experimental approach, methods of statistical analysis

The paper analyzes modern approaches to learning neural networks in simulated environments, in particular, the use of the Self-Play mode to automate learning. Also, the main problems and challenges related to reinforcement learning were considered, and ways to solve them were proposed.

The field of application is the game industry, application areas with the need for autonomous decision-making, and improvement of autonomous transport systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

RL – Reinforcement Learning (навчання з підкріпленням)

ML – Machine Learning (машинне навчання)

DL – Deep Learning (глибоке навчання)

PPO – Proximal Policy Optimization (оптимізація близької політики)

SAC – Soft Actor-Critic (м'який актор-критик)

ML-Agents – Unity Machine Learning Agents (агенти машинного навчання Unity)

API – Application Programming Interface (інтерфейс прикладного програмування)

ВСТУП

Імітаційні середовища стали ключовим інструментом у розвитку сучасних технологій, дозволяючи віртуально моделювати реальні ситуації з високою точністю. Ці середовища забезпечують контрольовані умови для проведення експериментів, досліджень та тестувань, що є критично важливим для різних галузей науки і техніки. Вони створюють можливість для детального аналізу складних систем, розробки і вдосконалення алгоритмів, а також оптимізації процесів без ризику для реальних об'єктів та ресурсів.

Розвиток штучного інтелекту і машинного навчання революціонізує багато аспектів сучасного життя, охоплюючи різноманітні сфери від ігрових додатків до автомобільної промисловості. Однією з передових галузей в цьому контексті є навчання з підкріпленням, яке займається розробкою алгоритмів, здатних до самонавчання через взаємодію з динамічним середовищем та отримання від нього зворотнього зв'язку.

Платформи розробки інтерактивних симуляцій набирають популярність в індустрії відеоігор та для наукових досліджень, особливо у сфері штучного інтелекту. Це пов'язано з їхньою здатністю моделювати різноманітні сценарії та віртуальні середовища, що забезпечує ідеальні умови для проведення експериментів та досліджень у сфері навчання з підкріпленням. Такі платформи дозволяють детально аналізувати та тестувати поведінку та стратегії штучних агентів у контрольованих умовах, що сприяє більш глибокому розумінню динаміки та ефективності навчальних алгоритмів.

Робота сконцентрована на дослідженні результатів різних методів та підходів навчання з підкріпленням у контрольованих імітаційних середовищах, з використанням платформи Unity як основного інструменту для створення цих середовищ. Особлива увага приділяється методам самонавчання та взаємодії агентів в різних умовах, а також впливу різних конфігурацій середовища на ефективність тренування нейронних мереж.

Актуальність дослідження обумовлена високим потенціалом застосування навчених моделей в реальному світі, включаючи робототехніку, автономні транспортні засоби, ігрову індустрію та інші сфери, де важливо забезпечити швидку адаптацію до умов, що змінюються, і самостійність ухвалення рішень. Вивчення та розвиток методів навчання з підкріпленням у імітаційних середовищах дозволить покращити алгоритми штучного інтелекту, зробити їх більш ефективними та універсальними для вирішення широкого спектру завдань, а в деяких випадках повністю замінити штучний інтелект.

Результатом роботи є система для машинного навчання з розробленеми контрольованими середовищами для порівняння результатів та оцінки якості. До результатів також можна віднести навчені моделі, що будуть порівнюватися між собою для виявлення залежності між налаштуваннями та якістю навчання за результатами тестування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність використання імітаційних середовищ

Імітаційні середовища стали невід'ємною частиною сучасних наукових досліджень і технологічних розробок. Вони дозволяють створювати детальні моделі реальних ситуацій у віртуальному просторі, що дає змогу проводити експерименти, навчання та тестування в умовах, які важко або небезпечно відтворити в реальному світі.

Імітаційні середовища існують у цифровому світі, що дозволяє моделювати різноманітні сценарії – від фізичних процесів до соціальних взаємодій. Віртуальність таких середовищ забезпечує можливість створення і контролю над умовами експериментів з максимальною точністю, налаштовуючи параметри відповідно до потреб дослідників. Це означає, що можна моделювати майже будь-яку ситуацію, від симуляції роботи складного механізму до відтворення взаємодії між людьми в різних соціальних умовах. Цифровий характер таких середовищ дозволяє швидко і легко змінювати параметри та умови, що забезпечує гнучкість і адаптивність досліджень [1].

Контрольованість є великою перевагою таких середовищ, що дає змогу встановлювати і змінювати умови для тестування різних моделей і алгоритмів. Це дозволяє створити ідеальні умови для випробувань, що сприяє отриманню надійних і відтворюваних результатів. Це дозволяє дослідникам проводити точні експерименти і отримувати достовірні дані, які можна повторно використовувати для подальших досліджень.

Використання віртуальних середовищ мінімізує ризики, пов'язані з реальними умовами, захищаючи як людей, так і обладнання. Це особливо важливо у випадках, де реальні експерименти можуть бути небезпечними або надзвичайно дорогими. Наприклад, у медицині симуляції хірургічних операцій дозволяють лікарям тренувати свої навички без ризику для пацієнтів, а в

автомобільній промисловості можна тестувати поведінку безпілотних автомобілів в екстремальних умовах без ризику для життя водіїв і пішоходів. Безпечність також означає, що можна проводити експерименти, які були б неможливі або неприйнятні в реальному світі через етичні або правові обмеження.

Можливість адаптації до різних рівнів складності – від простих моделей до складних систем з великою кількістю змінних – робить їх універсальним інструментом для досліджень. Масштабованість означає, що можна розширювати або зменшувати модель в залежності від потреб дослідження. Це дозволяє дослідникам починати з простих моделей і поступово додавати складності, перевіряючи, як нові змінні впливають на результати. Такий підхід особливо корисний у випадках, коли необхідно вивчити вплив великої кількості факторів на систему.

Імітаційні середовища забезпечують можливість повторного проведення досліджень в однакових умовах. Це критично важливо для перевірки і підтвердження результатів, що є основою наукової достовірності. Відтворюваність означає, що експерименти можуть бути точно повторені іншими дослідниками, що допомагає верифікувати результати і підвищує надійність досліджень. Це також дозволяє вносити зміни в експериментальні умови і спостерігати за тим, як ці зміни впливають на результати, що є ключовим аспектом наукового методу [2].

1.2 Актуальність дослідження підходів навчання нейронних мереж

Нейронну мережу можна визначити як метод штучного інтелекту, який вчить комп'ютери обробляти дані таким же способом, як і людський мозок. Це тип процесу машинного навчання, що називається глибоким навчанням, який використовує взаємопов'язані вузли або нейрони в шаруватій структурі, що нагадує людський мозок. Він створює адаптивну систему, за допомогою якої комп'ютери навчаються на своїх помилках та постійно вдосконалюються. Таким

чином, штучні нейронні мережі намагаються вирішувати складні завдання, такі як резюмування документів або розпізнавання осіб з більш високою точністю.

Штучний інтелект, методом якого є нейронна мережа – це широке поле, яке стосується використання технологій для створення машин і комп'ютерів, які мають здатність імітувати когнітивні функції, пов'язані з людським інтелектом, наприклад здатність бачити, розуміти та реагувати на усну чи письмову мову, аналізувати дані, давати рекомендації тощо.

Хоча штучний інтелект нерідко розглядається як система сама по собі, це набір технологій, реалізованих у системі, щоб дозволити їй міркувати, навчатися та діяти для вирішення складної проблеми [2].

Нейронні мережі використовуються в машинному навчанні, яке відноситься до категорії комп'ютерних програм, які навчаються без певних інструкцій. Зокрема, нейронні мережі використовуються в глибокому навчанні – вдосконаленому типі машинного навчання, яке може робити висновки з немаркованих даних без втручання людини. Наприклад, модель глибокого навчання, побудована на основі нейронної мережі та надана достатні навчальні дані, може ідентифікувати елементи на фотографії, яких вона ніколи раніше не бачила (рис. 1.1).

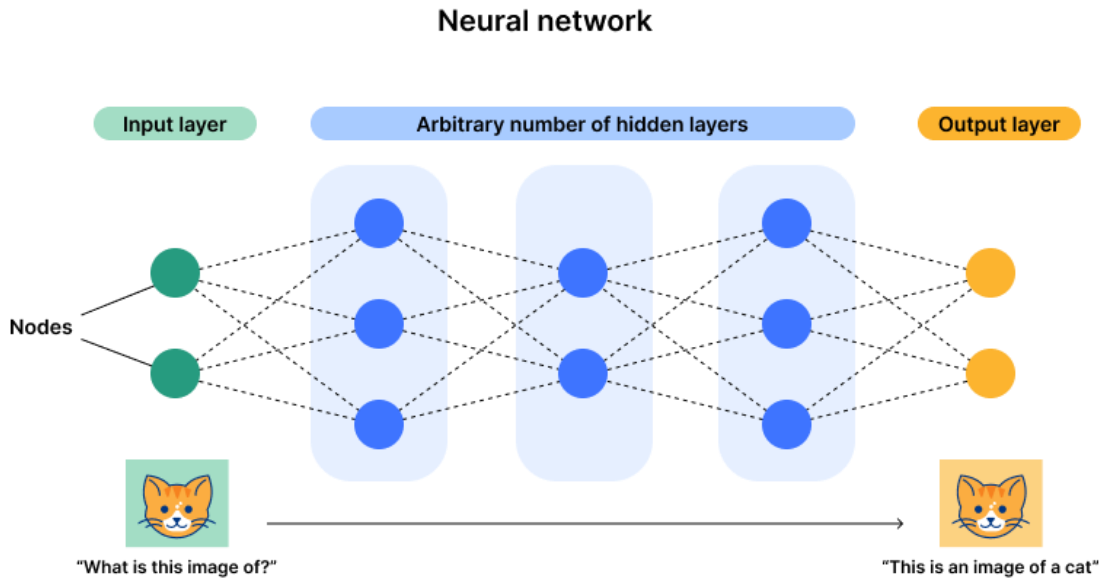

Рисунок 1.1 – Схематичне представлення нейромережі

Нейронні мережі складаються з набору вузлів. Вузли розподіляються принаймні на три шари:

– Вхідний шар

– «Прихований» шар

– Вихідний шар

Ці три шари є мінімумом. Нейронні мережі можуть мати більше одного прихованого шару, крім вхідного та вихідного.

Незалежно від того, до якого рівня він входить, кожен вузол виконує певне завдання або функцію обробки будь-яких вхідних даних, які він отримує від попереднього вузла (або від вхідного рівня). По суті, кожен вузол містить математичну формулу, причому кожна змінна у формулі має різну вагу. Якщо результат застосування цієї математичної формули до входу перевищує певний поріг, вузол передає дані на наступний рівень нейронної мережі. Якщо вихідні дані нижчі за порогове значення, дані не передаються на наступний рівень [3].

У сучасному світі, де інформаційні технології проникають в усі сфери життя, важливість нейронних мереж важко переоцінити. Ця актуальність зумовлена кількома важливими факторами:

– Великі обсяги даних: Сучасна цифрова епоха характеризується експоненціальним зростанням кількості даних, що генеруються людьми, машинами та бізнес-процесами. Нейронні мережі є одним з найефективніших інструментів для аналізу, обробки та інтерпретації таких даних завдяки своїй здатності виявляти складні залежності та взаємозв'язки у великих масивах даних;

– Темпи розвитку технологій штучного інтелекту: Штучний інтелект і машинне навчання розвиваються дивовижними темпами, стаючи все більш доступними та інтегрованими в багато сфер нашого життя. Нейронні мережі лежать в основі багатьох сучасних систем штучного інтелекту, від автономних транспортних засобів до персоналізованих рекомендаційних сервісів;

– Глобалізація та взаємопов'язаність: У світі, де кордони стають все більш невидимими, а дані циркулюють з дивовижною швидкістю, зростає потреба в

автоматизованій обробці та аналізі інформації. Нейронні мережі дозволяють автоматизувати та оптимізувати ці процеси, забезпечуючи швидку та ефективну обробку даних;

– Попит на персоналізацію: У сучасному світі споживачі вимагають персоналізації продуктів і послуг. Нейронні мережі дозволяють компаніям краще розуміти потреби та вподобання своїх клієнтів, адаптувати свої пропозиції та забезпечувати кращий користувацький досвід [4].

Якщо розглядати більш конкретні приклади застосування, то можна пригадати наявність автомобілів з автопілотом(рис 1.2), які відкривають не лише нові перспективи в індустрії, але й покращують рівень безпеки.

Рисунок 1.2 – Самокерований автомобіль

Використання такою допоміжної системи дозволяє виключити людський фактор, через який трапляється більшість дорожньо-транспортних пригод (втрата уваги, перевтома або порушення правил). Автомобілі з автопілотом

можуть значно знизити кількість таких інцидентів, оскільки вони не піддаються втомі, не відволікаються і завжди дотримуються правил дорожнього руху. Важливо відмітити, що допомога у боротьбі з перевтомою не лише допоможе уникнути небезпечною ситуації, але й підвищити загальну ефективність людини після поїздки.

Другим доволі схожим прикладом, але не дуже популярним у нашій країні, є роботи тихохідні роботи доставки (рис. 1.3) – компактні транспортні засоби на шести колесах, які рухаються суто по тротуарах із лімітованою швидкістю до 6 км/год. Це штучне обмеження, яке пов’язане з тим, що в локаціях, де вони дозволені, роботів фактично прирівняли до пішоходів і наділили такими самими правами з погляду правил дорожнього руху. Один із піонерів цієї індустрії, Starship Technologies, працює в кампусі Arizona State University в місті Тампі. Тут роботи, які доставляють їжу з ресторанів чи товари з магазинів – абсолютно буденна річ. Вони усюди: пересуваються студентськими кварталами, чекають на зелене світло на пішохідному переході або наступного замовлення поблизу кав’ярні [5].

Рисунок 1.3 – Роботи доставки

Безпілотні автомобілі, як і роботи доставки, можуть навчатися керувати в умовах, які можуть бути небезпечними для тестування в реальному світі. Імітаційні середовища дозволяють моделювати різні дорожні ситуації, включаючи екстремальні погодні умови, густий трафік та аварійні ситуації, що допомагає вдосконалювати системи керування і підвищувати безпеку автономних транспортних засобів.

Мабуть найкращим розповсюдженим додатком, що допомагає у програмуванні є ChatGPT – приклад трансформерної моделі, заснованої на архітектурі GPT (Generative Pre-trained Transformer). Трансформери - це тип архітектури нейронної мережі, що використовує механізми уваги для покращення можливостей моделі обробляти послідовні дані, наприклад текст.

GPT, розроблений OpenAI, представляє собою серію моделей (рис. 1.4), попередньо навчених на великих корпусах тексту. Ці моделі навчаються на

задачі прогнозування наступного слова в тексті, що дозволяє їм згенерувати текст, продовжуючи запропоновані початку фрази.

Однією з головних переваг ChatGPT є його здатність адаптуватися до різних контекстів і завдань. Від генерації коду до пояснення складних концепцій програмування, ChatGPT може допомогти розробникам швидко знаходити рішення і вдосконалювати свої навички. Завдяки попередньому навчанню на великому обсязі даних, модель здатна розуміти різні мови програмування і надавати релевантні поради та приклади.

Використання ChatGPT у програмуванні знижує час, необхідний на вирішення проблем, і підвищує ефективність розробки. Модель може допомагати у виправленні помилок, генерувати коментарі до коду, а також надавати рекомендації щодо оптимізації та кращих практик. Це робить ChatGPT незамінним інструментом для програмістів, сприяючи швидкому розвитку і впровадженню нових технологій.

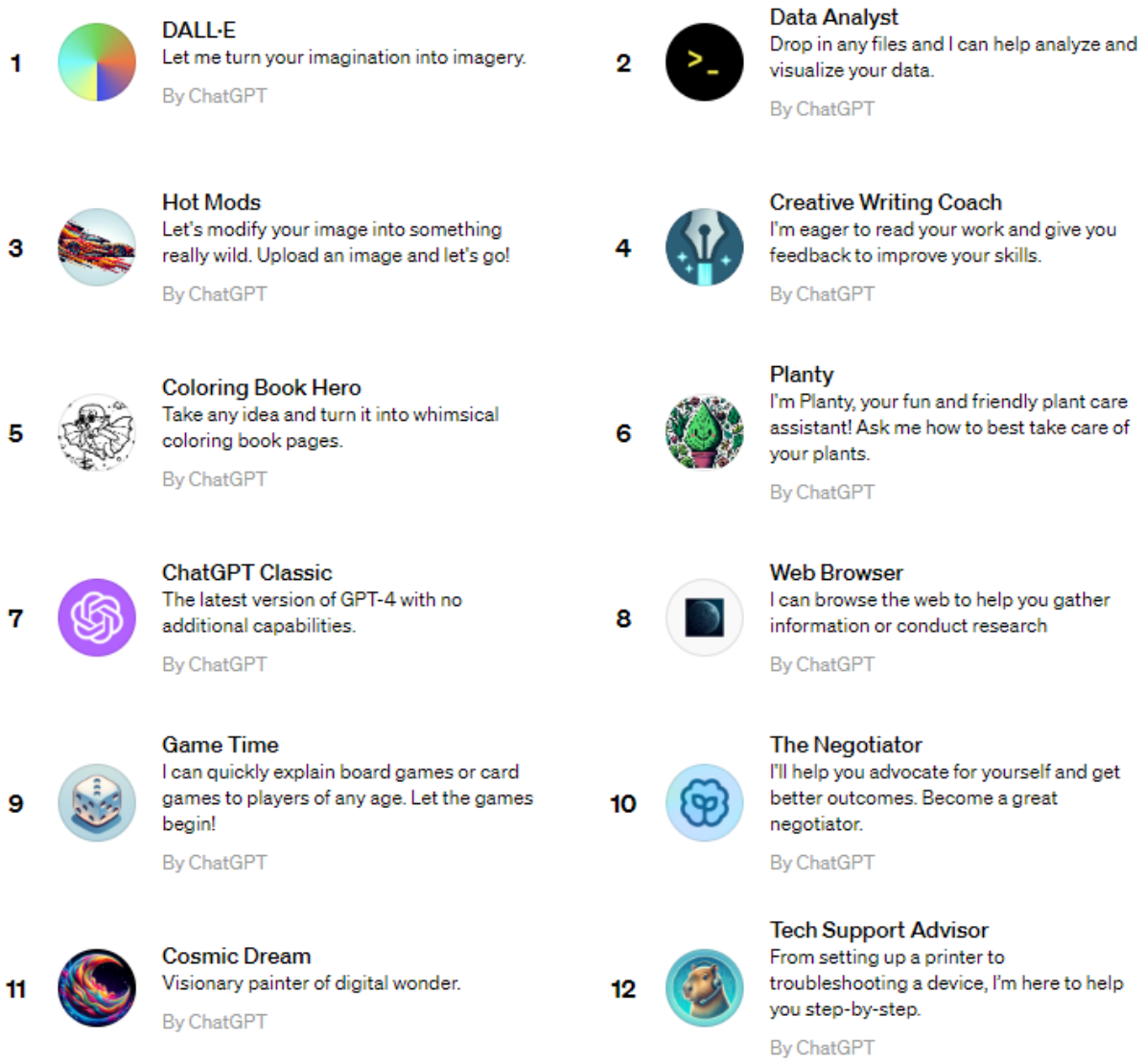

Рисунок 1.4 – Серія моделей від команди GPT

У контексті машинного навчання ChatGPT використовує концепцію навчання з учителем і без учителя. При попередньому навчанні модель не отримує конкретних інструкцій щодо правильності своїх відповідей, що є прикладом навчання без учителя. У фазі дообучення, коли модель адаптується до специфічних завдань або набору даних, які використовуються для навчання з учителем, моделі надають конкретні приклади з правильними відповідями.

Трансформерні моделі, такі як ChatGPT, продемонстрували значні досягнення в області природної мови, забезпечуючи більш глибоке розуміння контексту та здатність до більш природної та зв'язної генерації тексту. Це стало

можливим завдяки їх здатності обробляти довгі послідовності даних і усувати складні залежності від мови [6].

Неоціненний вклад нейромережі і штучний інтелект зробили для ігровою індустрії. За останні роки інтеграція штучного інтелекту та машинного навчання змінила ігровий ландшафт і спосіб взаємодії гравців з іграми. Один з їхніх основних внесків полягає в покращенні інтелекту характеру та аналізі поведінки. Обидві технології роблять не ігрових героїв та ігрове середовище розумнішими та персоналізованими для гравців. Моделі машинного навчання аналізують емоції гравців за допомогою голосу та рухів обличчя та розпізнавання форм, біометричних датчиків тощо. Практика пропонує більш чуйну взаємодію, персонажів емоційно розумними. Персонажі в грі також навчаються взаємодії з гравцями та експериментувати з різними стилями гри, застосовуючи нові стратегії та демонструючи людську поведінку.

Керовані штучним інтелектом персонажі будуть використовуватися в тренувальних симуляціях, щоб відобразити реальні сценарії. Ці симуляції призведуть до адаптації дій слухачів і забезпечать цінний зворотний зв’язок для навчання та розвитку навичок. Штучний інтелект також підтримує реалістичну анімацію та прийняття рішень NPC, збагачуючи сюжетний аспект ігор і наближаючи віртуальні світи до реальності.

Таким чином, важливість нейронних мереж у сучасному суспільстві зумовлена унікальними властивостями, здатністю долати сучасні виклики в обробці та аналізі даних, а також в можливості автоматизації повсякденних речей. Вони відіграють важливу роль у розвитку технологій і мають великий потенціал для подальшого прогресу в різних галузях і секторах.

1.3 Порівняння штучного інтелекту та машинного навчання

У той час як штучний інтелект охоплює ідею машин, які можуть імітувати людський інтелект, машинне навчання цього не робить. Машинне навчання має

на меті забезпечити точні результати, навчаючи машини виконувати певні завдання та виявляючи закономірності.

Штучний інтелект і машинне навчання – це не зовсім одне й те саме, але вони тісно пов'язані між собою. Необхідно трохи детальніше ознайомитися з відмінностями між цими технологіями:

– сфера застосування та функціональність: штучний інтелект охоплює ширший спектр, спрямований на відтворення людського інтелекту в машинах для виконання завдань у різних сферах.

Розглянемо приклади застосування. Популярним вже довгі часи є використання ШІ для управління промисловими роботами(рис. 1.5). Промислові роботи мають можливість контролювати власну точність і продуктивність, а також відчувати або виявляти, коли потрібне технічне обслуговування, щоб уникнути дорогих простоїв. Він також може діяти в новому або невідомому середовищі.

Рисунок 1.5 – Приклад промислових роботів

Іншим більш сучасним прикладом штучного інтелекту є інструменти персонального помічника, які є гаджетами для взаємодії людини та ШІ. Найпопулярнішими персональними помічниками є Google Home від Google, Siri від Apple, Alexa від Amazon і Cortana від Microsoft (рис. 1.6).

Рисунок 1.6 – Персональні асистенти

Ці особисті помічники дозволяють користувачам дізнаватися інформацію, бронювати готелі, додавати події до календарів, відповідати на запитання, планувати зустрічі, надсилати повідомлення чи електронні листи тощо.

З іншого боку, машинне навчання є підмножиною штучного інтелекту, що зосереджується саме на алгоритмах і статистичних моделях, які дозволяють системам навчатися та робити прогнози на основі даних. Як приклад, не можна не згадати рекомендації, які з'являються у виді рекламних інтеграцій на великій кількості веб-ресурсів (рис.1.7).

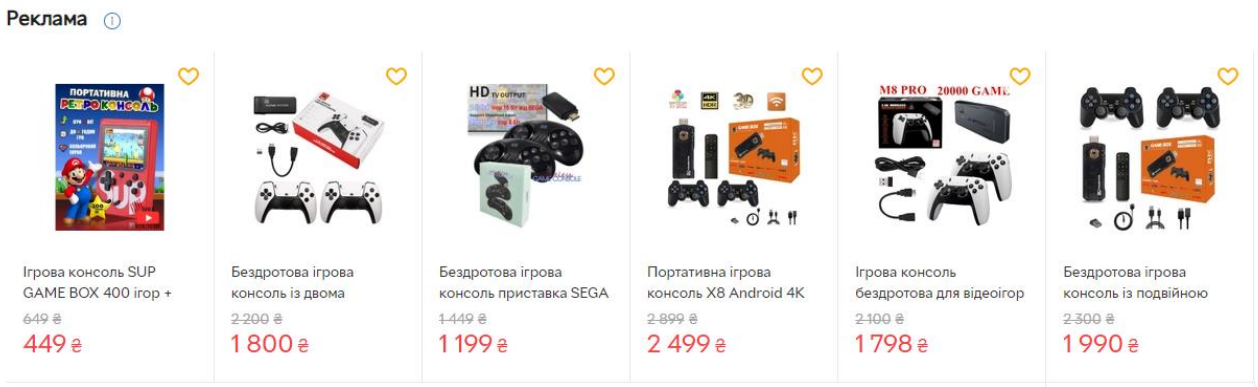

Рисунок 1.7 – Приклад реклами на основі історії переглядів

– залежність від даних. Машинне навчання значною мірою покладається на дані для навчання алгоритмів і підвищення продуктивності. Якість і кількість даних значно впливають на точність і ефективність моделей машинного навчання. Штучний інтелект, хоча він може використовувати дані, не завжди потребує великих наборів даних для своїх функцій; він може покладатися на заздалегідь визначені правила або логіку для виконання завдань;

– адаптивність і навчання: системи штучного інтелекту можуть включати як компоненти ML, так і компоненти, що не є ML. Однак ML особливо підкреслює здатність систем навчатися та вдосконалюватися на основі досвіду без явного програмування для кожного завдання. Моделі ML адаптуються та розвиваються на основі нових даних, постійно вдосконалюючи свої результати.

Якщо згадувати перетин і застосування технологій, то штучний інтелект і машинне навчання перетинаються в різних програмах. Системи штучного інтелекту часто включають методи машинного навчання, щоб розширити свої можливості. Галузі використовують алгоритми машинного навчання для прогнозної аналітики, розпізнавання зображень, обробки природної мови, систем рекомендацій тощо, сприяючи прогресу в охороні здоров’я, фінансах, автономних транспортних засобах і персоналізованому досвіді користувачів. Головною перевагою машинного навчання залишається ситуація, коли неможливо описати роботу штучного інтелекту, випадки, коли немає зрозумілого алгоритму для досягнення результату, або неможливо створити штучний інтелект, який зможе поводитися менш прогнозовано.

1.4 Постановка задачі дослідження підходів навчання з підкріпленням нейронних мереж у контрольованих імітаційних середовищах

Дослідження охоплює проектування та розробку застосунку з контрольованими імітаційними середовищами для навчання нейромереж. Такий підхід допомагає вирішити дві проблеми: швидкість розробки та фінансові витрати. Чому це важливо? Це вирішує одразу 2 проблеми: швидкість розробки

та фінансові витрати. Яким чином? Для навчання нейромережі необхідно надати умови, в яких вона буде взаємодіяти з оточенням, але в цьому є проблема, бо оточення може формуватися до самого кінця гри, тож після завершення розробки необхідно буде лише починати навчання нейромережі. Але чи настільки важлива повна завершеність середовища? Бо є випадки, коли це складніше організувати, коли мова йде не про ігрові застосунки, а про керовані механічні засоби, такі як автомобілі або роботи, що доставляють товари. Їм необхідно навчатися у складних ситуаціях з навантаженим трафіком, що є небезпечним і, через це, неможливим для реалізації. Полігон може не покривати усіх випадків, а у критичних ситуаціях може страждати піддослідна техніка. В таких випадках більш привабливим є варіант розробки середовища для навчання.

Задачі дослідження можна поділити на декілька пунктів:

– проаналізувати сучасні підходи до навчання з підкріпленням, особливості їх застосування в нейронних мережах, визначити ключові параметри для ефективного навчання;

– розробити критерії для вибору або створення контрольованих імітаційних середовищ, що найбільш адекватно відображають реальні умови або конкретні сценарії використання, а також середовища, на яких будуть проводитися навчання;

– розробити та імплементувати алгоритми навчання з підкріпленням, адаптовані до вибраних імітаційних середовищ.

– провести серію експериментів для оцінки ефективності навчених моделей в еталонному середовищі;

– проаналізувати отримані результати, виявити залежності між налаштуваннями середовища та якістю навчання і визначити оптимальні умови для навчання моделей;

– оцінити можливості застосування розроблених моделей в реальних сценаріях, включаючи робототехніку та автономні транспортні системи.

Серед очікуваних результатів можна відмітити кількісні та якісні показники ефективності методик навчання з підкріпленням у контрольованих

імітаційних середовищах, визначення оптимальних параметрів для покращення процесу навчання та адаптації, а також формулювання рекомендацій для застосування розроблених методів у різних областях, включаючи промисловість та наукові дослідження.

Розробка критеріїв буде досить важливим кроком, бо важливо витримати баланс у підході до навчання у системі та врахувати наступні параметри:

1) складність середовища. Необхідно змоделювати еталонне середовище, яке буде достатньо складним для звичайного штучного інтелекту і при цьому збереже адекватну швидкість навчання для моделей, які і будуть у ньому порівнюватися після навчання;

2) визначення даних для агентів. Необхідно виявити важливі для навчання елементи та ігнорувати зайві для пришвидшення навчання. До таких даних відносяться:

– стани: стан описує поточний стан справ у середовищі. Визначення того, які аспекти середовища важливі і мають бути включені до стану, є ключовим для ефективного навчання.

– дії: необхідно визначити, які дії доступні агенту. Від того, які дії можуть бути здійснені, залежить стратегія, яку агент зможе вивчити.

– нагороди: функція нагороди є центральним елементом навчання, оскільки вона направляє навчання агента, вказуючи, які дії кращі. Правильне визначення функції нагороди має вирішальне значення успіху навчання.

– політика: стратегія, яку використовує агент для вибору дій на основі станів середовища. Оптимізація політики є основною метою нейронної мережі в межах даної роботи.

3) система оцінки. Для пошуку кращої моделі з навчених, необхідно визначити спосіб визначення «переможця». Тобто це має бути зрозуміла оцінка, за якою можна точно виявити модель, результати якої були кращими. Це відрізняється від нагороди агента, бо агентів може бути декілька у системі, або навіть робота у команді.

4) загальні методи навчання. Необхідно змоделювати ситуацію та наділити агентів максимально наближеними до реальності способами взаємодії. Дані, як отримує агент, мають бути досяжними як у реальному середовищі, так і у віртуальному. Якщо використовуються віртуальне середовище, то є доступ до таких даних, як положення об'єкту у просторі. В реальному світі не вийде просто отримати координати частини середовища, яка недосяжна для агента.

Після навчання та нейромережі необхідно буде перевірити кожну з них на якість навчання, бо не виключені ситуації з неможливістю навчання за заданими параметрами. Також необхідно зберігати проміжні результати, для порівняння якості навчання за схожий період часу.

2 ВИБІР ПІДХОДУ МАШИННОГО НАВЧАННЯ ТА АНАЛІЗУ РЕЗУЛЬТАТІВ НАВЧАННЯ

2.1 Огляд та аналіз розповсюджених методів машинного навчання

У цьому розділі розглянемо чотири основні методи машинного навчання: навчання з учителем, без учителя з частковим наглядом та навчання з підкріпленням в межах глибокого навчання. Кожен з цих підходів має свої особливості, застосування та обмеження, які важливо розуміти, щоб обрати найбільш ефективну стратегію навчання для поставленого завдання та наявних даних.

Глибоке навчання — це підсфера штучного інтелекту, заснована на штучних нейронних мережах.

Оскільки алгоритми глибокого навчання також потребують даних для навчання та вирішення проблем, ми також можемо назвати це підсферою машинного навчання. Терміни машинне навчання та глибоке навчання часто розглядаються як синоніми. Однак ці системи мають різні можливості.

На відміну від машинного навчання, глибоке навчання використовує багаторівневу структуру алгоритмів, яка називається нейронною мережею.

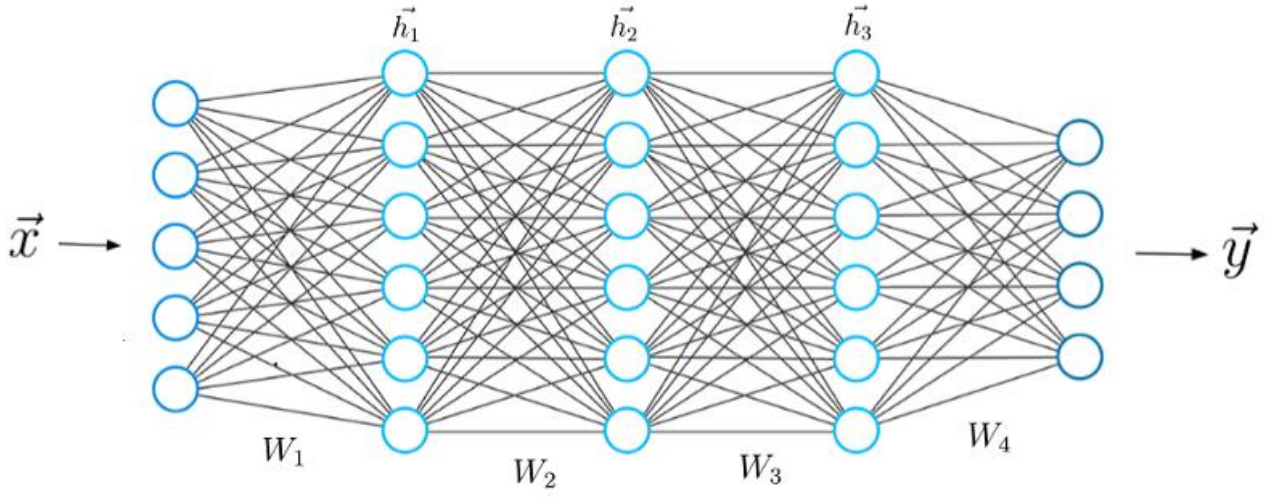

Рисунок 2.1 – Схематичне зображення нейронної мережі

Штучні нейронні мережі мають унікальні можливості, які дозволяють моделям глибокого навчання вирішувати завдання, які моделі машинного навчання ніколи не могли вирішити [7].

Тепер більш детально розглянемо кожен з методів. Навчання з учителем – це традиційний підхід у машинному навчанні, коли моделі навчаються на основі попередньо розмічених даних. Цей підхід передбачає, що існують чітко визначені вхідні та вихідні дані, що дозволяє моделі встановлювати відповідності між ними. Одне з найпоширеніших застосувань керованого навчання – класифікація, коли модель намагається визначити, до якої категорії належать вхідні дані. Іншим прикладом є регресія, яка визначає безперервні змінні на основі вхідних даних. Однак основними обмеженнями цього методу є те, що він вимагає великих обсягів маркованих даних і існує ризик надмірного перенавчання, коли модель дуже добре підходить до навчальних даних і втрачає здатність до узагальнення.

З іншого боку, навчання без учителя не вимагає маркування даних, оскільки модель намагається знайти структуру і взаємозв'язки в самих даних. Типовим застосуванням цього методу є кластеризація, де модель групує дані відповідно до їхніх характеристик. Інший приклад – виявлення аномалій, коли модель визначає відхилення від норми. Хоча навчання без учителя може впоратися з великими обсягами непозначених даних, воно страждає від складності правильної інтерпретації результатів і відсутності чітких критеріїв для оцінки ефективності.

Навчання з частковим наглядом поєднує в собі елементи двох описаних вище підходів і використовується, коли доступна лише обмежена кількість позначених даних. Цей метод ефективний в умовах, коли збір великої кількості позначених даних є затратним або неможливим. Навчання з частковим наглядом дозволяє використовувати великі обсяги непозначених даних разом з меншою кількістю позначених даних для покращення якості навчання моделі. Цей підхід може бути особливо корисним у таких сферах, як біологія та медицина, де збір точно позначених даних може бути складним та дорогим. Проте, одним з

викликів при навчанні з частковим наглядом є визначення оптимального співвідношення між позначеними та непозначеними даними, а також розробка алгоритмів, здатних ефективно використовувати обидва типи даних.

Кожен з цих методів машинного навчання має свої особливості та сфери застосування, а вибір конкретного методу залежить від особливостей завдання та наявних даних. Навчання під наглядом є найпоширенішим завдяки своїй наочності та простоті використання. Однак воно вимагає великих обсягів маркованих даних, що не завжди можливо. З іншого боку, неконтрольоване навчання пропонує унікальну можливість виявити приховані закономірності та взаємозв'язки в даних, але результати можуть бути складними для інтерпретації. Частково кероване навчання пропонує компроміс між цими двома підходами, забезпечуючи вищу точність моделі з меншою кількістю мічених даних.

Вибір цих підходів залежить від конкретного завдання і наявності даних. Тому детальне розуміння кожного підходу є ключем до успішного застосування машинного навчання в різних сферах.

Проте після ретельного розгляду особливостей та потенційних застосувань основних методів машинного навчання, особливу увагу було приділено навчанню з підкріпленням. Цей вибір був зумовлений кількома ключовими факторами, які відповідають специфіці дослідницької задачі, доступності даних та універсальності дослідження.

На відміну від навчання з учителем та без учителя, навчання з підкріпленням використовується для вирішення задач, де агент має взаємодіяти з динамічним середовищем, щоб досягнути певної мети. Агент навчається вибирати оптимальні дії на основі отриманого з середовища зворотного зв'язку, який зазвичай представлений у вигляді нагород. Такий підхід особливо релевантний для дослідницької задачі, яка включає розробку моделей для автономних систем, зокрема робототехніки, автономних транспортних засобів або ігрових додатків, де потрібно адаптуватися до змінних умов середовища.

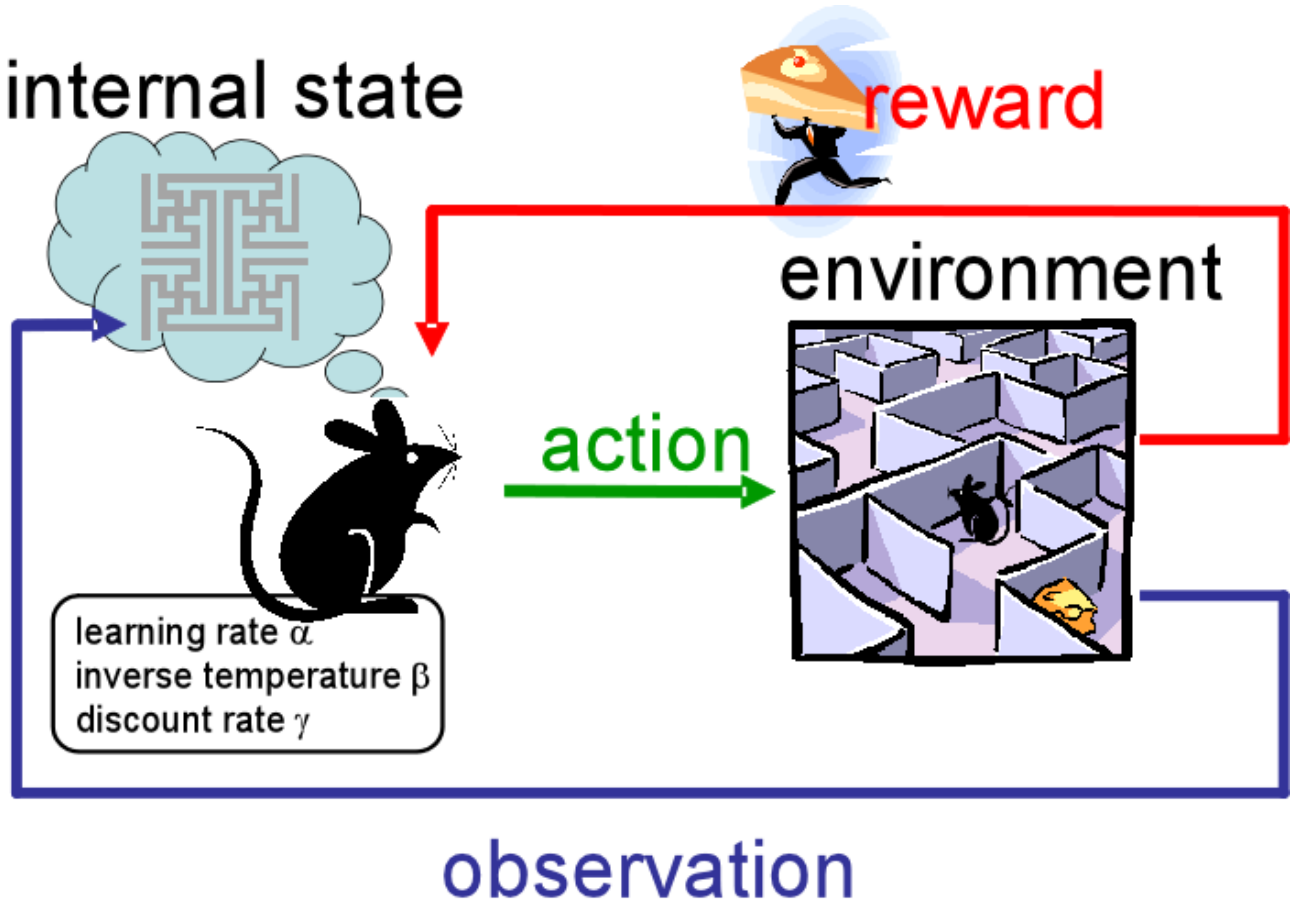

Рисунок 2.2 – Робота навчання з підкріпленням

Однією з головних переваг навчання з підкріпленням є його гнучкість та здатність до самовдосконалення через ітераційну взаємодію з середовищем. Це дозволяє моделі ефективно адаптуватися до складних та динамічних умов, що є критично важливим для реалізації ефективної поведінки автономних систем. Також, на відміну від інших методів, навчання з підкріпленням дозволяє моделі самостійно виявляти корисні стратегії поведінки, засновані не тільки на вже відомих даних, але й на нових досвідах, отриманих під час взаємодії з середовищем [8].

Враховуючи зазначені переваги та специфіку дослідницької задачі, було вирішено, що навчання з підкріпленням є найбільш відповідним методом. Цей підхід забезпечить можливість розробки моделей, здатних самостійно вчитися та адаптуватися в реальному часі.

2.2 Вибір методики та встановлення параметрів моделювання

При виборі методів машинного навчання для конкретного дослідницького завдання важливо звертати увагу на ряд критеріїв, щоб вибрати найбільш підходящий і ефективний підхід. Ці критерії є важливими для успішного впровадження та адаптації моделей машинного навчання в різних сферах застосування.

У контексті навчання з підкріпленням, аналіз даних набуває іншого виміру, оскільки традиційне розуміння даних тут зміщується від статичних наборів до динамічних середовищ, у яких агент взаємодіє. Такий аналіз включає оцінку наступних аспектів:

– реалістичність середовища: На відміну від звичайних датасетів, у навчанні з підкріпленням важливо, щоб середовище, в якому навчається агент, було максимально наближене до реальних умов застосування. Оцінка реалістичності включає перевірку параметрів середовища, їх варіативності та можливості моделювання реальних сценаріїв. При цьому необхідно розробити середовища для навчання, які матимуть схожу концепцію, але будуть відрізнятися від еталонного. Як цього можна досягти? Є 2 шляхи, які дозволять ввести різноманіття в середовища. Для першого варіанту необхідно змінювати кількість агентів, які взаємодіють із середовищем. Це допоможе змістити баланс вхідних даних від даних про агентів до даних про середовище. Другим шляхом є зміна середовища. Це допоможе зрозуміти наскільки важливо розробити саме точну копію середовища, для якого проводиться навчання агента.

– взаємодія з середовищем: Важливо оцінити механізми взаємодії агента з середовищем, зокрема, наявність та якість системи нагород і відповідей на дії агента. Ефективне навчання з підкріпленням вимагає чітко визначених і консистентних правил, що керують отриманням нагород.

– доступність середовища для експериментів: Необхідно забезпечити, що середовище доступне для повторних експериментів і тестування алгоритмів. Це

означає можливість його скидання до початкового стану, здатність моделювати різні ситуації та легкість інтеграції з різними алгоритмами навчання.

– адаптаційні можливості: Середовище для навчання з підкріпленням повинне дозволяти тестувати різні стратегії та підходи, адаптуючись до змін у поведінці агента. Важливо оцінити, наскільки гнучко середовище може відтворювати різноманітні умови та сценарії, які агент може зустріти.

– валідація та верифікація середовища: Перед початком тренувань агентів важливо перевірити, що середовище навчання відповідає всім вимогам та належним чином відображає реальні умови, для яких розробляється модель. Це допоможе забезпечити валідність та надійність отриманих результатів.

Ретельний аналіз цих аспектів дозволить ефективно підготувати основу для навчання з підкріпленням, забезпечивши наявність адекватного, релевантного та валідного середовища, яке допоможе досягнути поставлених цілей дослідження.

У рамках даної дослідницької роботи обрано розробку та використання спрощеної гри у футбол як середовища для навчання з підкріпленням. Вибір цього конкретного типу середовища зумовлений декількома факторами, які сприяють досягненню поставлених перед дослідженням цілей.

Зрозумілість призначення: Футбол є відомою та популярною грою, розуміння базових правил та цілей якої не викликає складнощів для більшості людей. Використання цієї гри як моделі для навчання з підкріпленням дозволяє легко ілюструвати мету навчання ботів – забивання голів і захист власних воріт, що робить результати дослідження зрозумілими та легко інтерпретованими.

Розповсюдженість футболу: Завдяки високій популярності футболу по всьому світу, використання цього виду спорту як бази для середовища дозволяє залучити увагу ширшого кола зацікавлених осіб, забезпечуючи легке розуміння досліджуваних процесів та стратегій.

Зрозумілість результатів: У футбольному матчі основним індикатором успіху є кількість забитих голів. Цей критерій є інтуїтивно зрозумілим і легко кількісно оцінюваним, що дозволяє чітко оцінити ефективність навчання агентів.

Можливість змінювати середовища: Спрощена гра у футбол надає широкі можливості для модифікації середовища, включаючи зміну кількості гравців, форми поля, а також додавання різноманітних обмежень чи перешкод. Це робить середовище гнучким та адаптивним до різних сценаріїв тестування та досліджень, дозволяючи вивчати вплив різних факторів на стратегію та поведінку навчених агентів.

Отже для спрощення тестування та фокусування на ключових аспектах навчання з підкріпленням, гра буде мати спрощений вигляд. Такий підхід дозволить зосередити увагу не на складності реалістичного моделювання повного футбольного матчу, а на вивченні базових принципів навчання з підкріпленням та взаємодії агентів з середовищем. Це важливо для того, щоб виявити фундаментальні закономірності та стратегії, які агенти можуть використовувати для досягнення успіху в заданому контексті.

Спрощення гри також сприятиме більш швидкому та ефективному процесу навчання, оскільки воно зменшує обчислювальні вимоги та спрощує процес аналізу результатів. Це особливо важливо для експериментальних умов, де потрібно проводити численні ітерації для тонкої настройки параметрів навчання та стратегій поведінки агентів, а також для умов, коли є залежність від ресурсів. За неможливості використовувати велику кількість ресурсів для навчання та при обмеженні часу необхідно скоротити складність навчання для агентів в імітаційних середовищах.

Вибір спрощеної гри у футбол як середовища для навчання також відкриває можливості для порівняння різних імітаційних середовищ. Змінюючи кількість гравців, форму поля та інші умови, можна дослідити, як різні середовища впливають на навчання та адаптацію агентів. Це дозволяє не тільки оцінити ефективність конкретних підходів до навчання з підкріпленням, але й зрозуміти, які фактори середовища найбільше впливають на успішність агентів.

Загалом, вибір спрощеної гри у футбол як середовища для навчання з підкріпленням забезпечує ідеальний баланс між зрозумілістю, адаптивністю та ефективністю для проведення дослідження, дозволяє зосередитися на

фундаментальних принципах та методиках навчання, замість того, щоб витрачати значні ресурси на візуалізацію та деталізацію, які не є критичними для основної мети дослідження. Це особливо важливо, враховуючи обмеженість обчислювальних ресурсів.

В контексті навчальних проектів та досліджень, обмеження обчислювальних ресурсів є типовим явищем. Високопродуктивні обчислювальні системи часто є дороговартісними та не завжди доступними для індивідуального користування. Тому вибір середовища, яке не вимагає значних обчислювальних потужностей, дозволяє проводити експерименти та тестування без необхідності використання великих серверних мощностей або спеціалізованого обладнання. Крім того, розробка та впровадження середовища для навчання з підкріпленням у рамках спрощеного сценарію дозволяє зосередитися на важливих аспектах навчального процесу, таких як вибір стратегій, оптимізація поведінки агентів та аналіз впливу різних параметрів середовища, не відволікаючись на обробку великих обсягів даних або управління складними графічними візуалізаціями.

Таким чином, вибір спрощеної гри в футбол як моделювального середовища для навчання з підкріпленням відповідає потребам ефективного використання обмежених обчислювальних ресурсів, одночасно забезпечуючи значущі та вимірювані результати дослідження.

2.3 Розробка методики дослідження отриманих результатів

У процесі розробки та оцінювання алгоритмів навчання з підкріпленням важливо мати збалансований набір метрик для всебічного аналізу продуктивності моделей. У випадку нашого дослідження, яке включає розробку спрощеного середовища футбольної гри, основні метрики, які будуть використовуватися для оцінки та порівняння результатів різних стратегій навчання, включають:

– Кількість забитих голів: Ця метрика є прямим індикатором успіху в спортивній грі і відображає здатність агента досягати кінцевої цілі - забити гол противнику.

– Кількість пройдених ітерацій: Вимірювання загальної кількості спроб, зроблених агентом для досягнення певних результатів, дає змогу оцінити витрати зусиль на навчання.

– Темп гри: Ця метрика відображає час, потрібний агенту для забиття гола від початку гри або відновлення гри після гола. Це дає уявлення про темп гри та ефективність атакуючих дій агента, а також дозволяє виявити випадковість забиття м'яча.

Аналіз цих метрик дозволяє отримати глибоке розуміння ефективності агентів у середовищі футбольної гри. Кількість забитих голів і час, за який ці голи були забиті, безпосередньо вказують на успіх агента у досягненні цілей гри. Водночас, оцінка часу навчання та кількості ітерацій надає інформацію про швидкість та ефективність навчального процесу. Врахування цих метрик у комплексі дозволить забезпечити об'єктивну оцінку продуктивності навчальних стратегій та вибрати оптимальний підхід.

3 ВИБІР ЗАСОБУ РОЗРОБКИ ТА РОЗРОБКА ДОДАТКУ

3.1 Обґрунтування вибору середовища та засобів розробки

Настав час перейти до етапу розробки і почати з вибору середовища моделювання, яке має вирішальне значення для створення необхідних умов тестування алгоритмів машинного навчання. Зокрема, навчання з підкріпленням вимагає, щоб агенти навчилися взаємодіяти зі складними середовищами і досягали цілей без прямих інструкцій, лише за допомогою системи винагород і покарань. Через велику варіативність потенційних завдань та високі вимоги до адаптивності агентів, вибір імітаційного середовища, яке може ефективно моделювати реальність та забезпечувати можливість зміни сценаріїв, стає ключовим фактором успіху дослідження.

У цьому контексті, Python визнаний лідером серед мов програмування для реалізації проектів на основі нейронних мереж. Він відіграє ключову роль у цій сфері завдяки своїм багатим бібліотекам та фреймворкам, які спрощують складність моделювання та тренування мереж. Ось декілька аспектів, які роблять Python особливо цінним для роботи з нейронними мережами:

– Спеціалізовані бібліотеки: Python підтримує широкий спектр бібліотек для нейронних мереж, таких як TensorFlow, PyTorch, і Keras. Ці бібліотеки надають готові до використання високорівневі абстракції та низькорівневі оперативні інструменти, які дозволяють ефективно розробляти, тренувати та валідувати моделі глибинного навчання [9].

– Гнучкість у моделюванні: Завдяки бібліотекам, які надають велику свободу у створенні архітектур мереж, Python є ідеальним для експериментування з новітніми дослідницькими ідеями у галузі нейронних мереж. Він дозволяє легко модифікувати шари, активаційні функції, та втрати функції, забезпечуючи науковцям необхідні інструменти для інновацій.

– Сумісність та інтеграція: Бібліотеки Python для нейронних мереж можуть бути легко інтегровані з іншими науковими та інженерними інструментами, що дозволяє широко застосовувати машинне навчання у різних доменах. Також, їх легко використовувати разом з іншими API та зовнішніми системами, що розширює можливості імплементації моделей у реальних умовах.

Саме можливість інтеграції з зовнішніми системами буде використана в даній дослідницькій роботі. В якості такої системи обрано Unity – одну з провідних платформ для розробки ігор та імітаційних середовищ, що використовується в широкому спектрі досліджень, зокрема в області навчання з підкріпленням. Його вибір як основи для імітаційного середовища дослідження нейронних мереж обумовлений кількома ключовими перевагами:

– Багатофункціональність та візуалізація: Unity забезпечує потужні візуальні можливості, включаючи підтримку комплексної графіки та фізики, що дозволяє створювати реалістичні та деталізовані середовища. Це є особливо корисним у контексті навчання з підкріпленням, де точне відтворення фізичних властивостей середовища може суттєво вплинути на навчання агентів.

– Гнучкість у створенні сценаріїв: За допомогою Unity можна створювати майже необмежену кількість сценаріїв взаємодії агентів з середовищем. Це дає можливість не тільки імітувати існуючі умови, але й експериментувати з цілком новими, інноваційними ситуаціями. Така гнучкість є критично важливою для глибинного дослідження поведінки нейронних мереж.

– Інтеграція з Python: Unity може бути інтегрований з Python через різноманітні плагіни та API, що дозволяє використовувати всі переваги Python для програмування логіки нейронних мереж. Це включає можливість взаємодії між високорівневими алгоритмами навчання, розробленими на Python, та динамічними, візуально багатими середовищами Unity [10].

– Можливість масштабування та адаптації: Unity підтримує розробку як для настільних, так і для мобільних платформ, включаючи VR та AR, що робить його ідеальною платформою для розробки рішень, що можуть бути адаптовані під різні технологічні потреби та умови використання.

Ці властивості роблять Unity не тільки платформою для створення ігор, але й міцним інструментом для наукових досліджень у сфері штучного інтелекту, зокрема, для випробування та вдосконалення нейронних мереж у контрольованих імітаційних середовищах.

Відтепер постає питання зв'язування мови програмування Python та рушія Unity, який використовую мову C#. На допомогу приходить Unity Machine Learning Agents Toolkit (ML-Agents), який заслуговую окремою уваги.

3.2 Огляд Unity ML-Agents

Unity Machine Learning Agents Toolkit – це проект із відкритим вихідним кодом, який дозволяє іграм і симуляторам служити середовищем для навчання інтелектуальних агентів. Реалізація (на основі PyTorch) найсучасніших алгоритмів, щоб дозволити розробникам ігор і любителям легко навчати інтелектуальних агентів для ігор 2D, 3D і VR/AR. Дослідники також можуть використовувати наданий простий у використанні API Python для навчання агентів за допомогою навчання з підкріпленням, навчання імітації, нейроеволюції або будь-яких інших методів. Цих навчених агентів можна використовувати для багатьох цілей, включаючи контроль поведінки NPC (у різноманітних налаштуваннях, таких як багатоагентні та суперницькі), автоматизоване тестування ігрових збірок і оцінювання різних рішень щодо дизайну гри перед випуском. Набір інструментів ML-Agents є взаємовигідним як для розробників ігор, так і для дослідників штучного інтелекту, оскільки він забезпечує центральну платформу, на якій можна оцінити досягнення ШІ в багатих середовищах Unity, а потім зробити їх доступними для більш широких спільнот дослідників і розробників ігор [11].

Розглянемо, з чим ми маємо справу. З Unity ML-Agents у нас є чотири важливі компоненти (рис. 3.1):

– Навчальне середовище, яке містить сцену Unity та всіх персонажів гри. Сцена Unity забезпечує середовище, в якому агенти спостерігають, діють і

навчаються. Те, як можна налаштувати сцену Unity як навчальне середовище, насправді залежить від мети навчання. Наприклад, при намаганні вирішити конкретну проблему навчання з підкріпленням обмеженого масштабу, і в цьому випадку можна використовувати ту саму сцену як для навчання, так і для тестування навчених агентів. Або можна навчати агентів працювати в складній грі чи симуляції. У цьому випадку може бути більш ефективним і практичним створити спеціальну навчальну сцену. Набір інструментів ML-Agents включає SDK ML-Agents Unity, який дає змогу перетворювати будь-яку сцену Unity на навчальне середовище, визначаючи агентів та їхню поведінку.

– Python Low-Level API — містить низькорівневий інтерфейс Python для взаємодії та керування навчальним середовищем. На відміну від навчального середовища, API Python не є частиною Unity, а живе поза ним і спілкується з Unity через комунікатор. Цей API міститься у спеціальному пакеті Python і використовується в процесі навчання для зв’язку з академією та керування нею під час навчання. Однак його можна використовувати і для інших цілей. Наприклад, можна використовувати API для використання Unity як механізму моделювання для ваших власних алгоритмів машинного навчання.

– Зовнішній комунікатор – який з’єднує навчальне середовище з API низького рівня Python. Він живе в навчальному середовищі.

– Тренери Python, які містять усі алгоритми машинного навчання, які дозволяють тренувати агентів. Алгоритми реалізовано на Python і є частиною власного пакету mlagents Python. Пакет надає єдину утиліту командного рядка mlagents-learn, яка підтримує всі навчальні методи та параметри, викладені в цьому документі. Інтерфейс Python Trainers виключно з низькорівневим API Python [12].

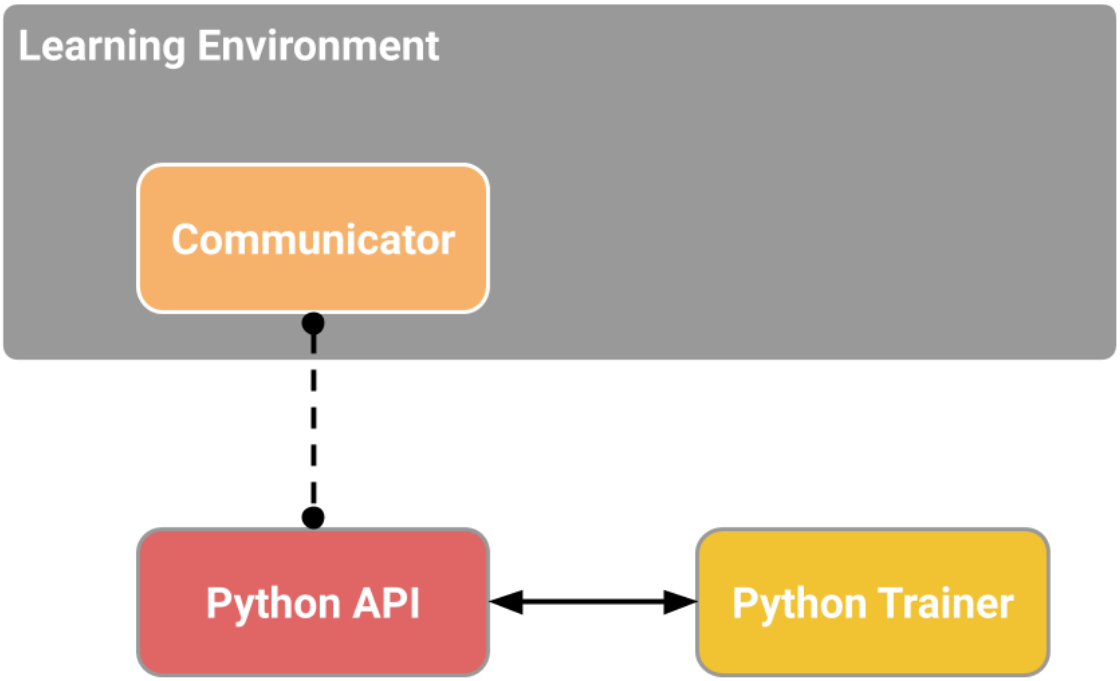

Рисунок 3.1 – Спрощена блок-схема ML-Agents

Всередині навчального компонента ми маємо різні елементи (рис. 3.2):

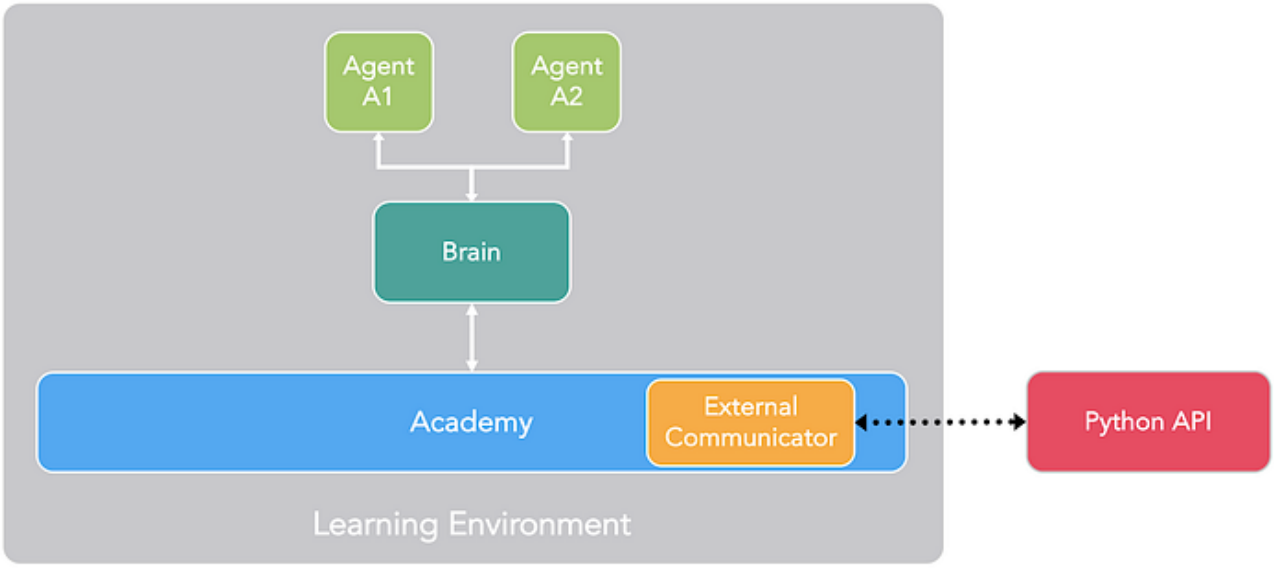

Рисунок 3.2 – Декомпозиція навчального середовища

Перший — Агент, актор сцени. Саме його ми збираємося навчити, оптимізувавши його політику (яка вкаже нам, які дії вживати в кожному стані) під назвою «Мозок».

Нарешті, є Академія, цей елемент керує агентами та процесом прийняття ними рішень. Академію – як майстер, який обробляє запити від API python.

Щоб краще зрозуміти його роль, пригадаємо процес RL. Це можна змоделювати як цикл, який працює так:

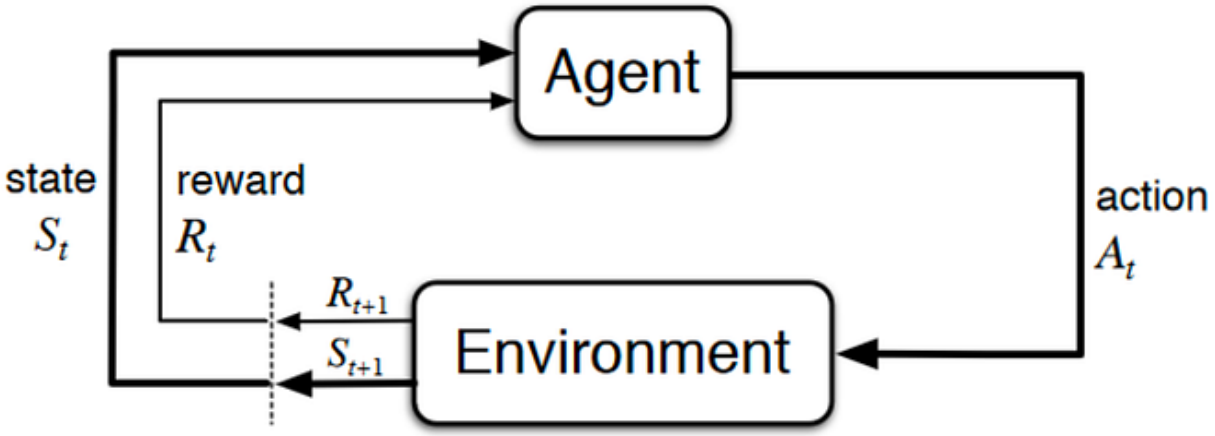

Рисунок 3.3 – Взаємодія агент-середовище в навчанні з підкріпленням [13]

Цей цикл навчання з підкріпленням виводить послідовність стану, дії та винагороди. Метою агента є максимізація очікуваної сукупної винагороди. Насправді Академія надсилатиме замовлення агентам і забезпечуватиме синхронізацію агентів [14].

Корисним доповненням є готова реалізація Self-play – технікою навчання з підкріпленням, що дозволяє агентам навчатися, змагаючись один з одним у контрольованому середовищі. Цей метод ґрунтується на принципі ітеративного суперництва між агентами, що призводить до поступового покращення їх навичок у міру того, як вони адаптуються до поведінки один одного. Ця методика ефективно використовується у завданнях, де необхідні складні стратегічні та тактичні рішення [15].

Self-play найбільш відомий своїм застосуванням у навчанні штучного інтелекту для ігор, таких як шахи та Go. Приклади включають AlphaGo та AlphaZero від DeepMind, які демонструють, як моделі, навчені з використанням self-play, можуть досягти та перевершити рівень гри найкращих людських гравців. У сценаріях, де важливе моделювання соціальної взаємодії або поведінки натовпу, self-play може допомогти агентам навчитися передбачати та реагувати на дії інших інтелектів. Також у розрахованих на багато користувачів іграх або завданнях, де агенти повинні працювати разом для досягнення спільної

мети або змагатися один з одним, self-play допомагає розробити ефективні стратегії взаємодії [16].

3.3 Розробка програмного засобу

Для початку розробки програмного засобу створено новий проект, для якого необхідно не лише завантажити рушій Unity, але й Python, PyTorch та власне ML-Agents. Встановлення останніх можна перевірити вже всередині Unity:

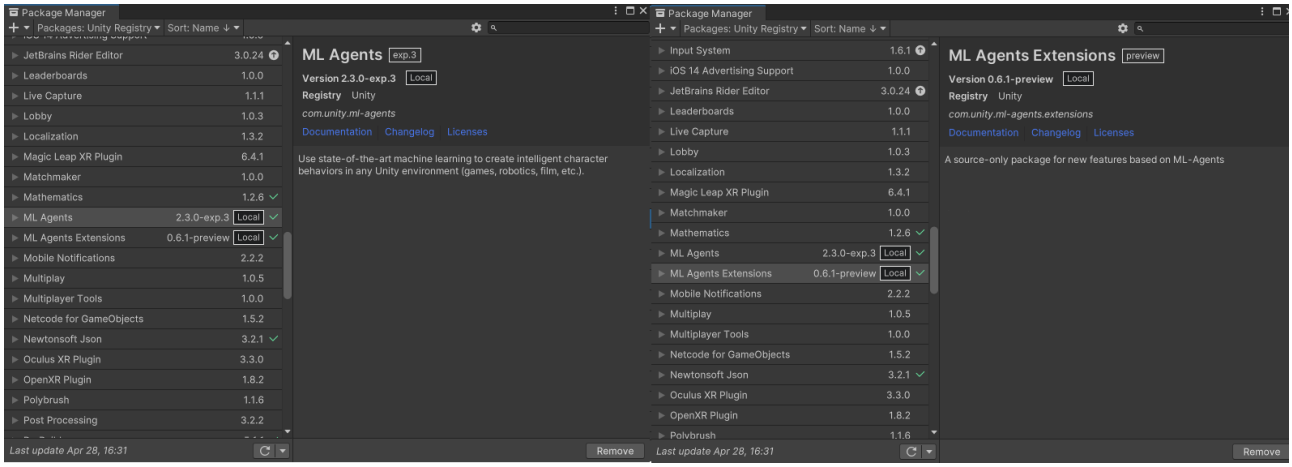

Рисунок 3.4 – Менеджер пакетів Unity

Перейдемо до створення тестового середовища. Для цього виділимо основні компоненти сцени [17]. Для гри у футбол це м'яч, футбольне поле та самі гравці. Усе середовище у свою чергу поділяється на наступні компоненти:

– Базовий шар (підлога), який не є необхідним, але дозволяє краще сприймати процес:

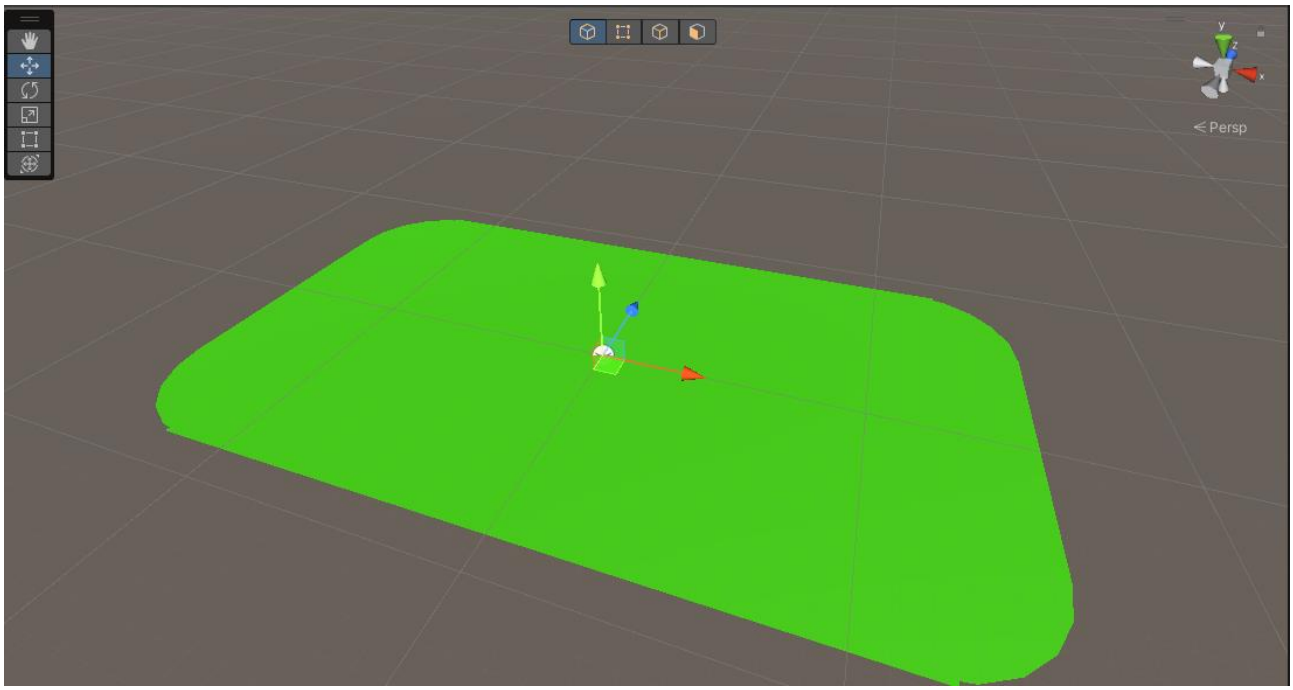

Рисунок 3.5 – Базовий шар

– Бар'єри, які допоможуть ідентифікувати межі ігрового поля:

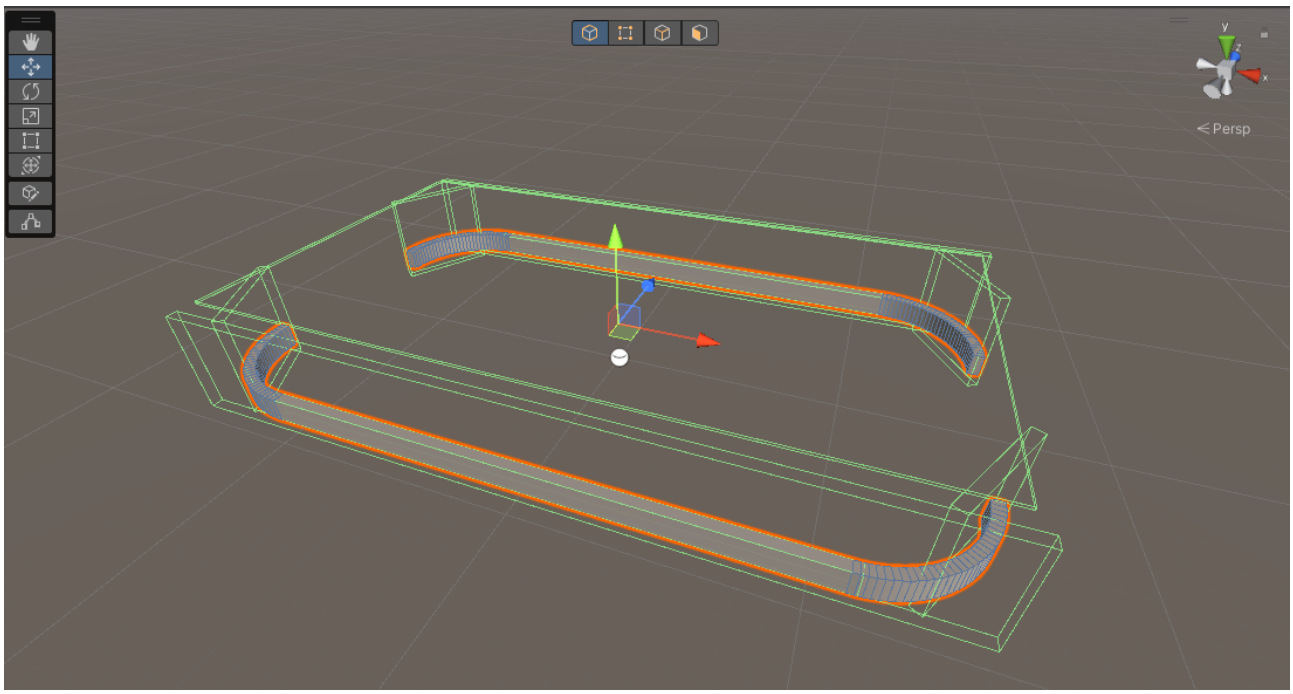

Рисунок 3.6 – Межі ігрового поля

– Ворота команд:

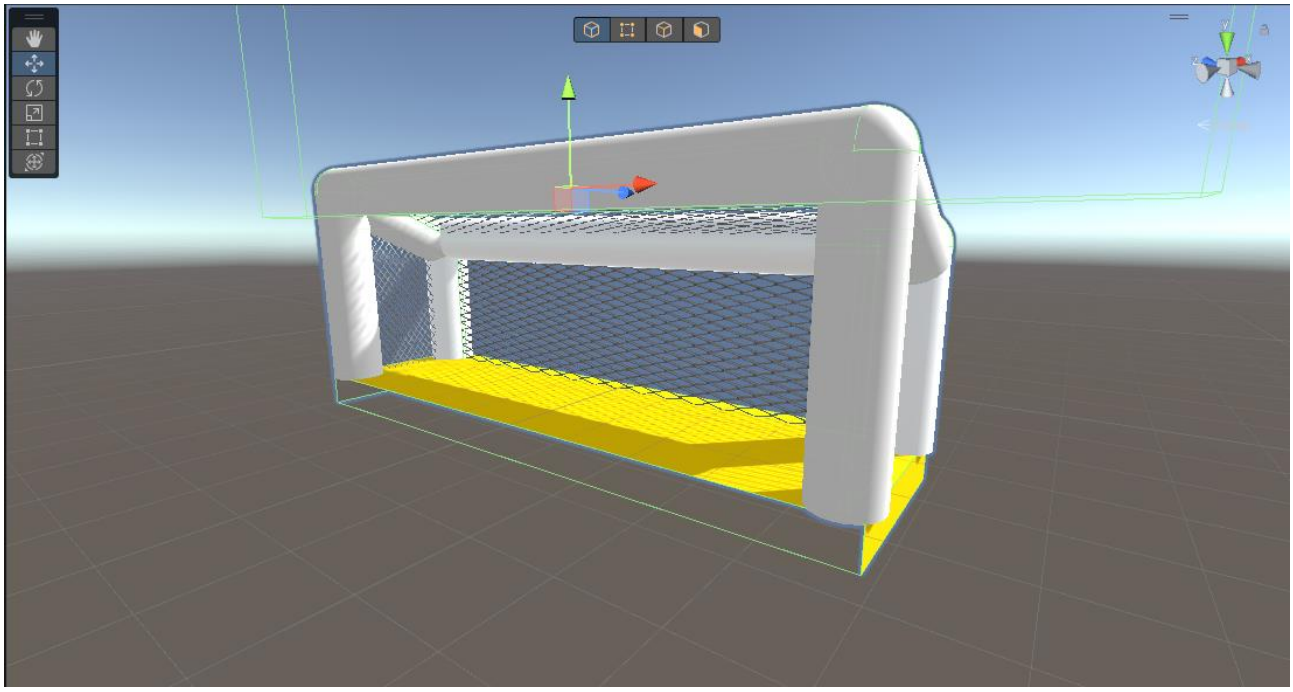

Рисунок 3.7 – Приклад воріт однієї з команд

– М'яч:

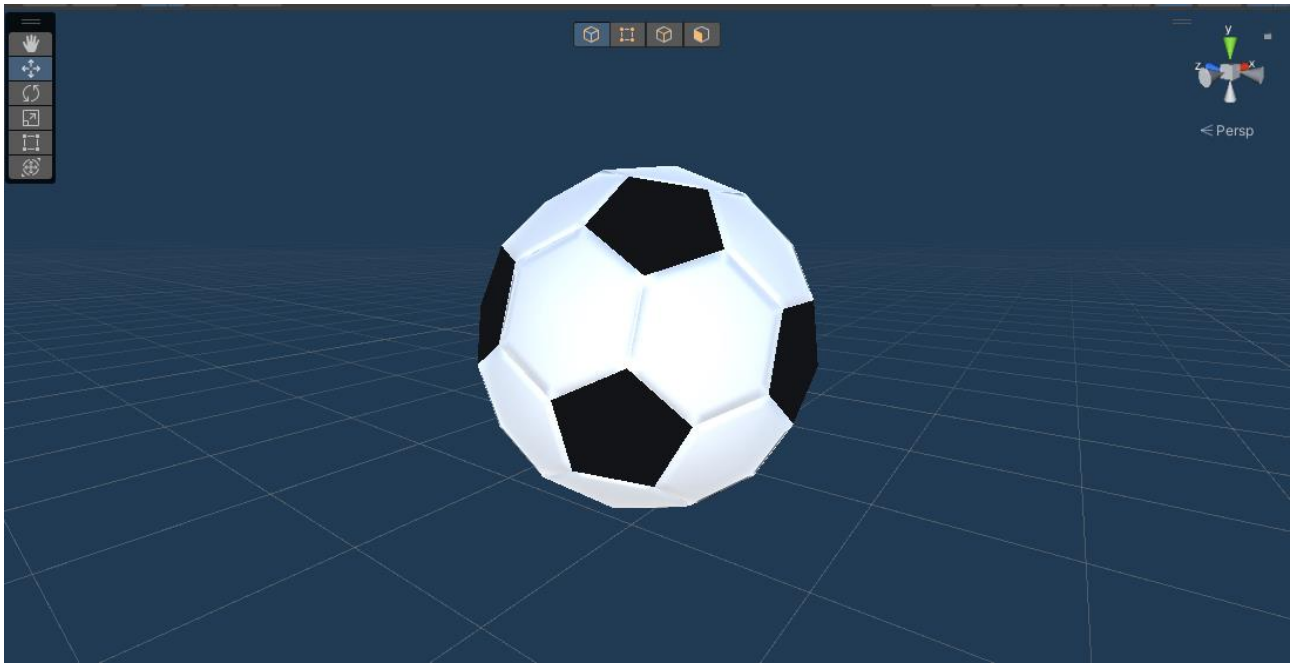

Рисунок 3.8 – М'яч

– Примітивні гравці:

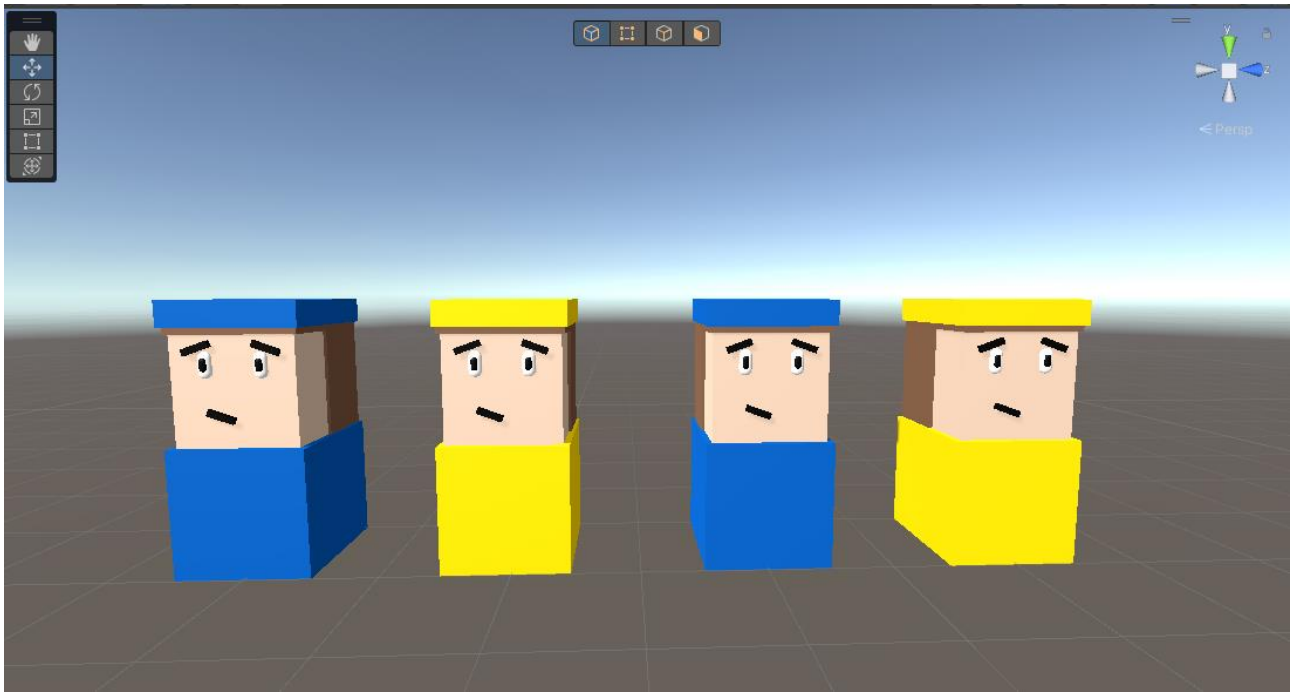

Рисунок 3.9 – Гравці різних команд

– Табло з часом та рахунками команд:

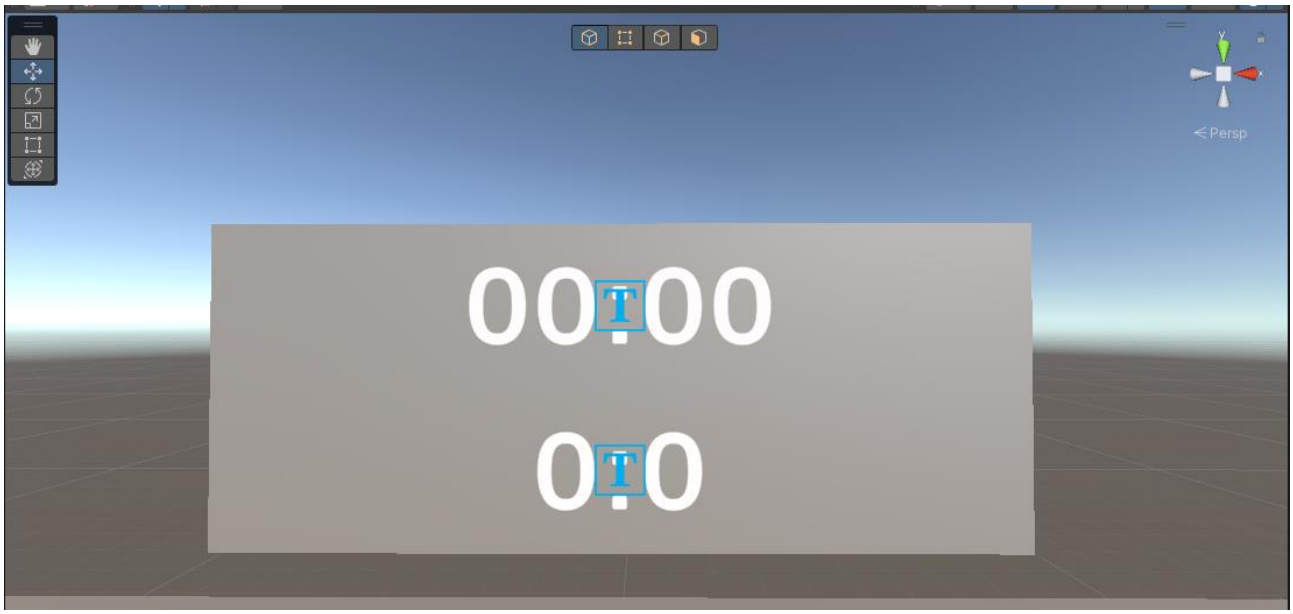

Рисунок 3.10 – Табло

Повноцінна картина одного з тестових середовищ має наступний вигляд:

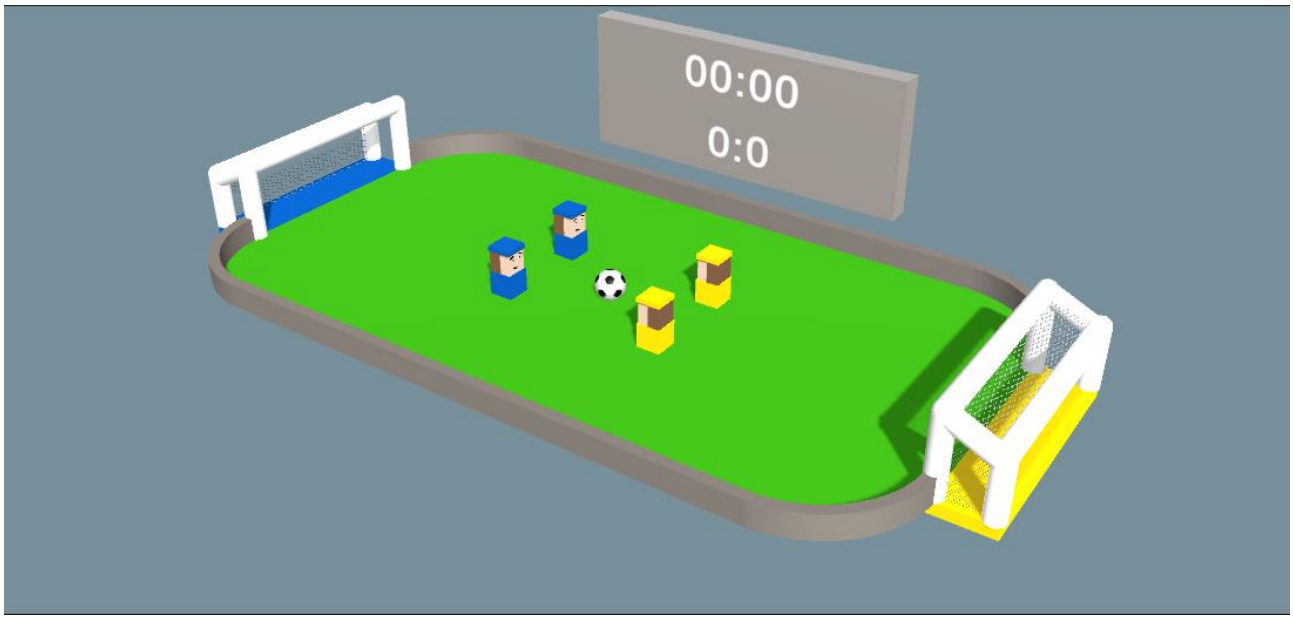

Рисунок 3.11 – Приклад тестового середовища

Після створення елементів середовища перейдемо до створення та налаштування агентів. Налаштування агентів також складається з декількох етапів.

Перш за все, кожного агента необхідна наділити можливостями, які б мав реальний гравець. У переміщенні середовищем не виникає проблем, агент має доступ до усіх необхідних варіантів рухів, а саме:

– рух уперед та назад;

– рух ліворуч та праворуч;

– обертання навколо осі.

Ці параметри повертаються в якості результату від нейронної мережі. Їх кількість та значення залежить від налаштування конкретної моделі агента:

Рисунок 3.12 – Налаштування вихідних даних

Тут можна встановити кількість виходів нейромережі та тип значення. Виходи поділяються на 2 типи:

– Безперервні дії – дозволяють агенту виконувати дії, які можуть набувати будь-якого значення в заданому безперервному діапазоні. Це корисно у сценаріях, де потрібне точне налаштування вихідних параметрів, а у нашому випадку необхідні конкретні величини;

– Дискретні дії – на відміну від безперервних дискретні дії обмежені набором конкретних, окремо взятих опцій. Цей тип дій зручний у середовищах, де агент повинен вибирати з кінцевого списку чітко визначених опцій.

Дискретних дій обрано три, для перелічених раніше варіантів руху. Кожна дія може мати 3 стани (приклад для руху): не роботи нічого (значення 0), іти вперед (значення 1), іти назад (значення 2).

Отримання інформації про положення в середовищі відбувається за допомогою сенсорів, які являють собою проміні та розходяться у різні боки від кожного агента:

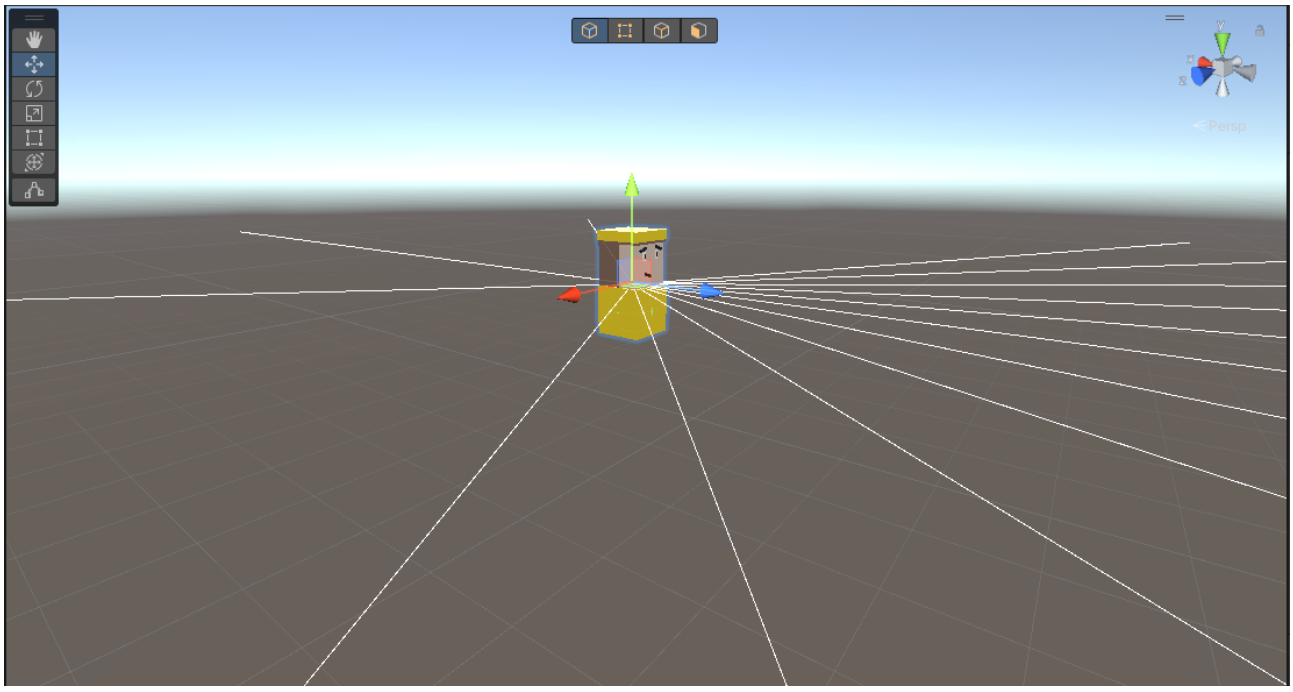

Рисунок 3.13 – Сенсори агента

При перетині з об'єктом, який є у переліку тегів, які можна виявити, передається на вхід нейромережі відстань до об'єкта для подальших розрахунків. Візуально виглядає наступним чином:

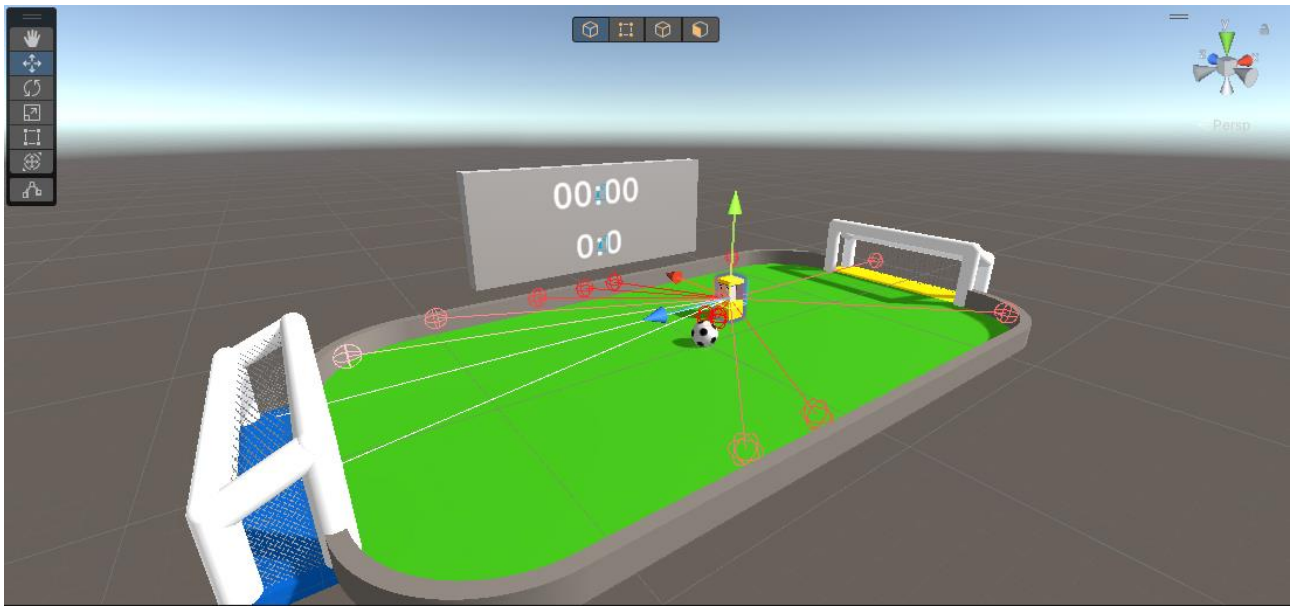

Рисунок 3.14 – Перетин сенсорів із об'єктами

Налаштування сенсорів відбувалося комбінацією реально можливого для людини куту огляду(передні сенсори) та можливості озирнутися(менша кількість сенсорів позаду). Кількість сенсорів було підібрано при перших спробах навчання нейромережі, а саме момент, коли збільшення кількості не вливало на швидкість та якість навчання:

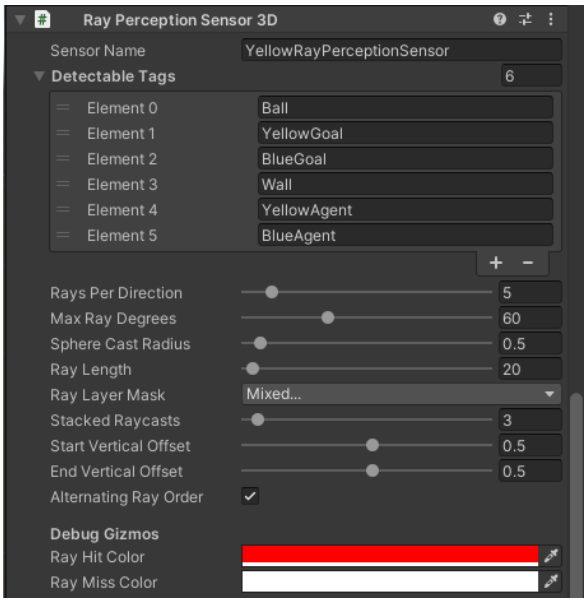

Рисунок 3.15 – Приклад налаштування сенсорів

Усі агенти мають однакові налаштування вхідних та вихідних даних у навчальному середовищі для більш коректного порівняння впливу середовища на навчання.

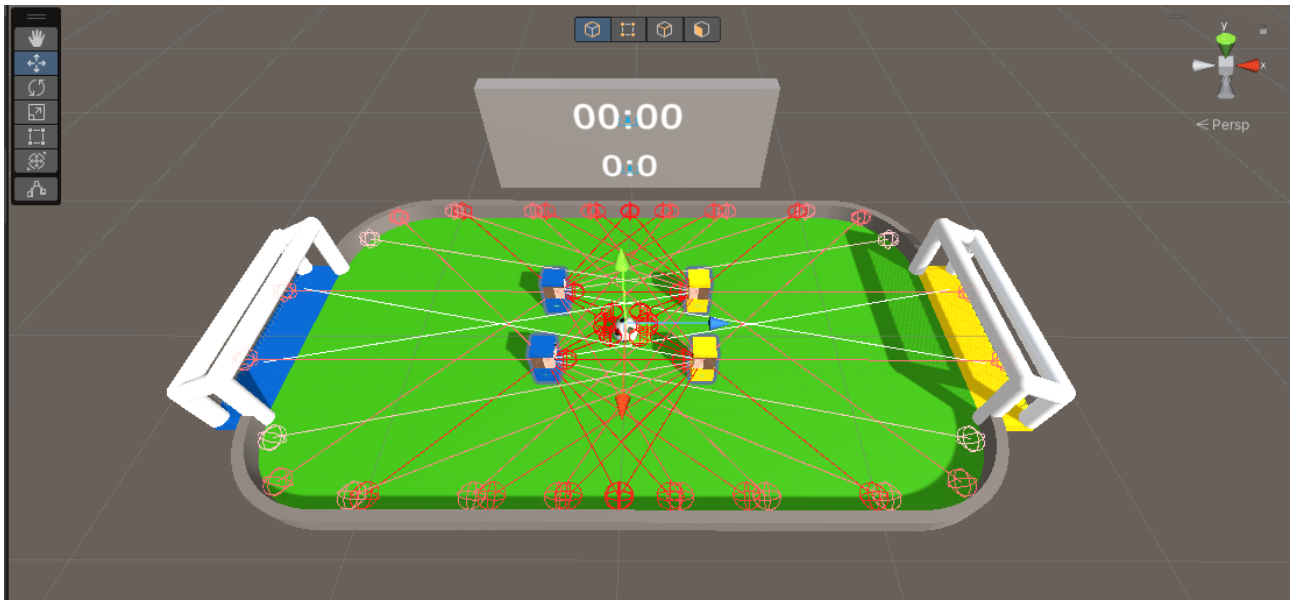

Рисунок 3.16 – Доступна вхідна інформація агентів

Розглянемо наступний важливий аспект навчання з підкріпленням – систему винагород агентів. Встановлення винагороди відбувається лише в одному випадку досягнення мети гри – м'яч потрапив у ворота. При цьому, діють різні правила винагороди для команд:

– команда, якій забили гол: отримує винагороду у розмірі -1;

– команда, що забила гол: отримує винагороду залежно від часу, який їй знадобився для того, щоб забити гол.

Розрахунок часу вимірюється наступним чином: в юніті реалізовано метод, який фіксовано визивається 50 разів на секунду і при кожному визові додається одиниця до кількості кроків.

Тестове середовище обмежено по кроках до 5000. Це зроблено для того, аби агенти не втрачали контроль та не завершували гру забиванням у кути ігрового поля. Після того, як досягнуто мети у 5000 кроків, проводиться скидання середовища до початкового стану. Це включає в себе повернення гравців на стартову точку з невеликим зміщенням, аби нейромережа не так сильно запам'ятовувала алгоритм дій у грі, а також скидання позиції м'яча у центр поля.

Тож, підбиваючи підсумки, винагорода для команди, яка забила гол, розраховується за наступною формулою:

$$\text{Винагорода} = 1 - \frac{\text{Кількість кроків}}{5000}$$

Формула 3.1 – Розрахунок винагороди

Розглянемо налаштування нейромережі для навчання. Файл конфігурації в ML-Agents для Unity відіграє критично важливу роль, оскільки він визначає параметри та умови, в яких агенти навчатимуться. Це схоже на створення правил та умов для експерименту у наукових дослідженнях. Основне завдання файлу конфігурації – забезпечити гнучкість та контрольованість процесу навчання без необхідності змінювати код самої програми.

Файл також відіграє важливу роль для збереження результатів, які будуть використані для навчання і порівняння якості навчання.

Перед початком обрано конфігурацію, яка буде еталонною для порівняння якості навчання (рис. 3.16). Інші конфігурації будуть виступати в якості модифікацій контрольованих імітаційних середовищ.

Шляхом випробувань для конфігурації 2 на 2 гравці було обрано наступні параметри:

Лістинг 3.1 – Файл конфігурації

```
behaviors:
  SoccerTwo:
    trainer_type: poca
    hyperparameters:
      batch_size: 2048
      buffer_size: 20480
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: constant
    network_settings:
```

```
    normalize: false
    hidden_units: 512
    num_layers: 2
    vis_encode_type: simple
  reward_signals:
    extrinsic:
      gamma: 0.99
      strength: 1.0
  keep_checkpoints: 500
  max_steps: 50000000
  time_horizon: 1000
  summary_freq: 10000
  self_play:
    save_steps: 50000
    team_change: 200000
    swap_steps: 2000
    window: 10
    play_against_latest_model_ratio: 0.5
```

Розглянемо біль детально параметри:

– hyperparameters.trainer_type – Тип тренажера для використання: ppo, sac або poca. Обрано саме poca, розроблений командою ML-Agents. Poca це багатоагентний алгоритм навчання, який може витончено впоратися з агентами породження та вимкнення без використання станів поглинання [18]. Обрано саме цей варіант, бо інші 2 не дуже добре працюють без гарного вхідного матеріалу;

– hyperparameters.batch_size – Кількість досвідів у кожній ітерації градієнтного спуску. На початку було обрано мінімально рекомендований варіант, але після експериментів значення підвищено до 2048;

– hyperparameters.buffer_size – Обсяг досвіду, який необхідно зібрати, перш ніж модель політики може бути оновлена. Відповідає обсягу досвіду, який необхідно зібрати, перш ніж модель може бути вивчена або оновлена. Значення корелює з batch_size та кратно перевищує його;

– hyperparameters.learning_rate – Початкова швидкість навчання градієнтного спуску. Відповідає інтенсивності оновлення на кожному кроці градієнтного спуску. Рекомендоване значення;

– hyperparameters.beta – Регулювання ентропії робить політику «більш випадковою». Це дозволяє агентам досліджувати простір дій належним чином під час навчання. Збільшення ентропії гарантує, що буде виконуватися більше випадкових дій. Рекомендоване значення;

– hyperparameters.epsilon – Впливає на швидкість еволюції політики під час навчання. Градієнт відповідає порогу толерантності до різниці між старими і новими політиками під час приземлення оновлень. Встановлення цього значення меншим призводить до більш стабільних оновлень, але також уповільнює процес навчання. Рекомендоване значення;

– hyperparameters.lambd – Параметр редагування, який використовується для обчислення узагальнених оцінок переваг. Його можна розуміти як ступінь, до якого агент покладається на поточну оцінку цінності при обчисленні оновленої оцінки цінності. Низькі значення відповідають більшій довірі до поточної оцінки цінності, тоді як високі значення відповідають більшій довірі до фактично отриманих винагород у середовищі. Рекомендоване значення;

– hyperparameters.num_epoch – Кількість проходів через буфер досвіду (batch_size) під час оптимізації градієнтного спуску. Рекомендоване значення;

– hyperparameters.learning_rate_schedule – Визначає, як швидкість навчання змінюється з часом. Linear зменшує швидкість навчання лінійно і досягає нуля при max_steps, в той час як constant зберігає швидкість навчання постійною протягом всього циклу навчання. Рекомендоване значення;

– network_settings.normalize – Чи застосовувати нормалізацію до вхідних даних векторних спостережень. Ця нормалізація базується на поточному середньому значенні та дисперсії векторних спостережень. Нормалізація корисна для складних неперервних задач керування, але шкідлива для простіших дискретних задач керування. Рекомендоване значення;

– network_settings.hidden_units – Кількість одиниць у прихованому шарі нейронної мережі. Вона еквівалентна кількості одиниць у кожному повністю пов'язаному шарі нейронної мережі. Для простих задач, де правильна дія - це проста комбінація спостережуваних входів, ця кількість повинна бути

невеликою. Воно повинно бути більшим для задач, де дія є дуже складною взаємодією між спостережуваними змінними. В нашому випадку обрано значення у 512 нейронів, що більше за рекомендоване;

– network_settings.num_layers – Кількість прихованих шарів у нейронній мережі. Це еквівалентно кількості прихованих шарів, присутніх після вхідних даних спостереження або після того, як візуальне спостереження було закодовано ШНМ. Для простих завдань менша кількість шарів, швидше за все, навчиться швидше і ефективніше. Для більш складних завдань управління потрібна більша кількість шарів. Рекомендоване значення;

– network_settings.vis_encode_type – Тип кодера для кодування візуальних спостережень.За замовчуванням використовується простий кодер, який складається з двох згорткових шарів. Рекомендоване значення;

– reward_signals.extrinsic.gamma – Коефіцієнт дисконтування для майбутніх винагород, отриманих від навколишнього середовища. Це можна розглядати як важливість віддалених майбутніх винагород для агента. Цей параметр має бути високим у випадках, коли агенту потрібно діяти зараз, щоб підготуватися до винагороди у віддаленому майбутньому. Якщо винагорода ближча, цей параметр буде нижчим. Рекомендоване значення;

– reward_signals.extrinsic.strength – Коефіцієнт, який множить винагороду, отриману від навколишнього середовища. Типові діапазони залежать від сигналу винагороди. Рекомендоване значення;

– keep_checkpoints – Максимальна кількість контрольних точок моделі для зберігання. Не впливає на навчання і буде використане для аналізу якості навчання агентів;

– max_steps – Загальна кількість кроків (тобто зібраних спостережень і виконаних дій), які необхідно завершити в одному середовищі (або в усіх середовищах, якщо паралельно використовується більше одного середовища), перш ніж процес навчання буде завершено. Для початку було обрано стандартне значення у 500.000 кроків, але після досягнення значення прийнято рішення про

збільшення кількості до 50.000.000 для порівняння результатів навчання за більший час та додаткового спостереження.

– time_horizon – Скільки кроків цінності досвіду потрібно зібрати для кожного агента, перш ніж його буде додано до буферу цінності досвіду. Якщо ця межа досягнута в кінці епізоду, оцінка вартості використовується для оцінки загальної очікуваної винагороди на основі поточного стану агента. Значення збільшено зі стандартних 64 до 1000, що поліпшило якість навчання.

– summary_freq – Кількість досвіду, необхідного для створення та перегляду статистики навчання [19]. Не впливає на якість навчання;

– self_play.save_steps – Кількість тренувальних кроків між знімками. Збільшення значення save_steps дає змогу охопити ширший діапазон супротивників з різними рівнями навичок та стилями гри, оскільки політика отримує більше тренувань. В результаті агенти тренуватимуться проти більшої кількості супротивників. Навчити політику перемагати більш різноманітних супротивників є складнішим завданням і може вимагати більш узагальнених тренувальних кроків, але може призвести до створення більш узагальненої та надійної політики в кінці навчання. Це значення також залежить від того, наскільки складним є середовище для агента. За результатами обрано значення 50.000, що є більшим за рекомендоване;

– self_play.team_change – Кількість кроків тренера до заміни команди тренерів. Це кількість кроків тренера, яку команда, пов'язана з певним тренером-привидом, тренуватиме до того, як інша команда стане новою тренувальною командою. Чим вище значення Зміни команди, тим довше агент може тренуватися проти опонента. Чим довше агент тренується проти одного суперника, тим більше суперників він може перемогти. Однак надто довге тренування може призвести до переробки стратегії певного супротивника, що може стати причиною невдачі агента проти наступної групи супротивників. Рекомендоване значення.

– self_play.swap_steps – Кількість кроків-привидів (не кроків тренера) між заміною політики супротивника на інший знімок. «Крок-привид» - це крок,

зроблений агентом, який не навчається за фіксованою політикою. Причиною такої відмінності є те, що в асиметричних іграх можуть бути команди з нерівною кількістю агентів: команда з двома агентами збирає вдвічі більше кроків агента за один крок середовища, ніж команда з одним агентом. Тому ці дві величини повинні бути різними, щоб обміняти однакову кількість кроків інструктора на однакову кількість кроків опонентів. Рекомендоване значення;

– self_play.window – Розмір плаваючого вікна минулих знімків, з якого вибираються супротивники агента. Як і у випадку з гіперпараметром save_steps, агент тренується проти більшої кількості супротивників. Вивчення політики для перемоги над більш різноманітними супротивниками є більш складною проблемою і може вимагати більш загальних кроків навчання, але може призвести до більш загальної та надійної політики наприкінці навчання. Рекомендоване значення;

– self_play.play_against_latest_model_ratio – Ймовірність того, що агент грає проти останньої політики опонента; більше значення play_against_latest_model_ratio вказує на те, що агент частіше грає проти поточного опонента. Коли агент оновлює свою політику, опоненти змінюються на кожній ітерації. Це може призвести до нестабільного середовища навчання, але також може змусити агента автоматично планувати складніші ситуації, роблячи фінальну політику жорсткішою. [20]. Рекомендоване значення.

Після усіх налаштувань можна переходити до навчання агентів. Під час навчання можна побачити статус навчання у консолі (рис. 3.17), який оновлюється раз на 10 тис. кроків:

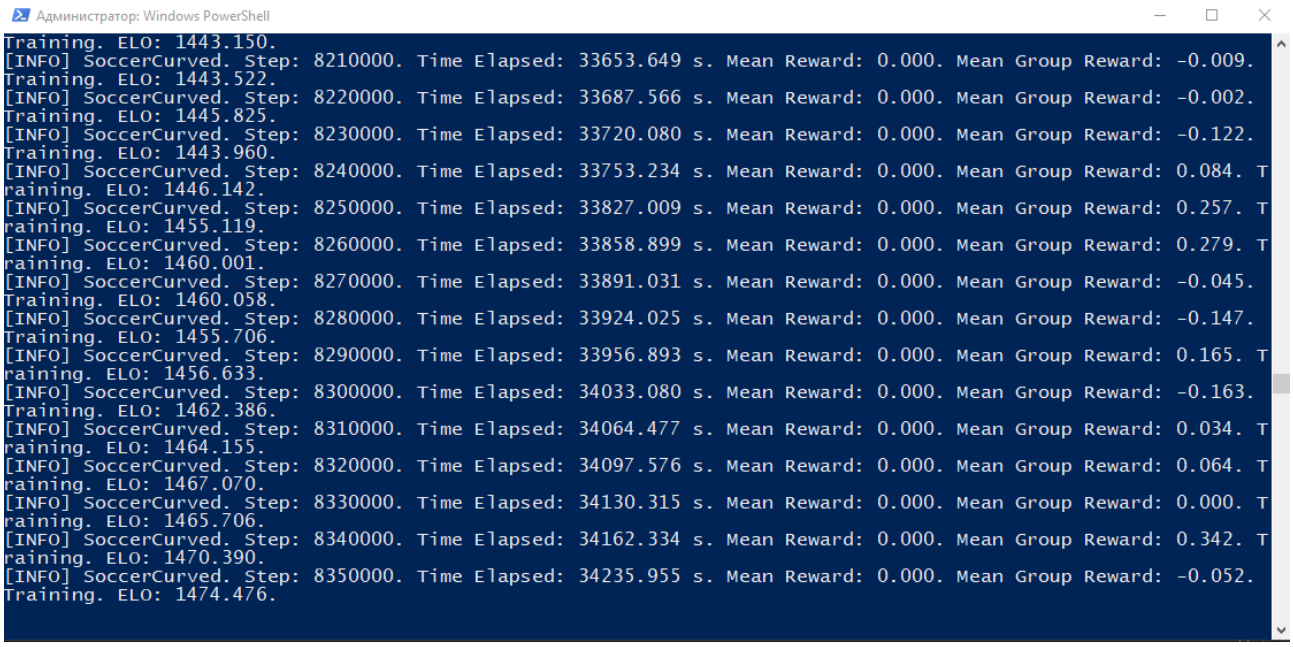

Рисунок 3.17 – Інформація про стан навчання

Можна побачити наступну інформацію: поточна кількість кроків (Step); час після минулого чекпоінту (Time Elapsed); середня винагорода (Mean Reward), яка не має значення у випадку поточної політики винагород; середня групова винагорода (Mean Group Reward) – не є надто показовим полем, бо команди змагаються одна з одною, але допомагає помітити, що навчання не проходить вдало; Оцінювання ігрових навичок агента (ELO) – особистий числовий рейтинг агента.

На даному етапі проведено усі налаштування середовища та конфігурації нейромережі, настає час довгого тренування агентів в імітаційних середовищах для подальшого порівняння якості.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ЕКСПЕРИМЕНТУ

Для проведення порівняння якості навчання агентів в імітаційних середовищах було розроблено декілька сцен, в яких проводилося тренування. Кожна сцена мала свої унікальні параметри та конфігурації, що дозволяло оцінити різні аспекти навчання агентів. Як зазначалося раніше, буде декілька етапів порівняння, кожен з яких має своє важливе значення для оцінки загальної ефективності:

1) Загальна якість навчання:

– Кількість забитих голів: Цей показник безпосередньо відображає успіх агента у досягненні основної мети - забити м'яч у ворота противника. Висока кількість забитих голів свідчить про ефективність стратегії та вміння агента використовувати свої навички.

– Співвідношення забитих і пропущених голів: Цей параметр дозволяє оцінити збалансованість гри агента. Агент, який забиває багато голів, але також багато пропускає, може мати агресивну, але неефективну в довгостроковій перспективі стратегію;

2) Швидкість навчання – Порівняння тих самих параметрів загальної якості, але через певну кількість ітерацій: Цей етап дозволяє побачити, наскільки швидко агент вчиться. Час, за який агент досягає високих показників, є важливим критерієм ефективності алгоритму навчання. Швидке навчання свідчить про те, що алгоритм ефективно використовує ресурси та здатен швидко адаптуватися до середовища.

3) Темп гри – Час, за який буде досягнуто мети: Важливо не тільки досягати цілей, але й робити це ефективно з точки зору часу. Цей параметр допомагає визначити, наскільки швидко агент може приймати рішення та виконувати дії у грі, що є критичним для реальних застосувань, де швидкість реакції може бути вирішальною [21].

Першим етапом буде перевірка якості навчання у конфігурації 2 на 2 гравці, яку обрано за відправну еталонну точку. Саме з цією моделлю буде проходити порівняння інших моделей. Вибір саме цього варіанту імітаційного середовища є цілком випадковим, для прикладу можна обрати будь-яке інше середовище. Конфігурація 2 на 2 дозволяє створити збалансоване середовище, яке є достатньо складним для демонстрації ефективності агентів, але водночас не надто перевантажує систему, що дозволяє швидко отримувати результати та аналізувати їх.

Для початку використаємо моделі нейромереж, що пройшли навчання протягом 50 мільйонів ітерацій, та почекаємо, поки буде забито тисячу голів, аби побачити, яку різницю можна отримати при використанні однакового розуму для агентів. Цей підхід дозволяє мінімізувати вплив випадкових факторів та забезпечує об'єктивне порівняння результатів. Очікується, що моделі зможуть демонструвати різні стратегії та підходи до гри, що допоможе виявити сильні та слабкі сторони кожного підходу.

Ця похибка також допоможе у майбутньому порівнювати різницю у навчанні. Окрім цього, важливо врахувати, що кількість ітерацій та забитих голів може варіюватися залежно від складності середовища та початкових умов. Таким чином, ці дані стануть основою для подальших експериментів, дозволяючи налаштовувати параметри навчання для досягнення оптимальних результатів [22].

Кожному агенту встановлено модель з 50 мільйонами ітерацій (рис. 4.1). У налаштуваннях також встановлено обмеження у тисячу голів для зупинки процесу. Для прискорення процесу отримання результатів встановлено параметр масштабу часу на значення 10, для прискорення сцени у 10 разів. Також для полегшення підрахунку часу додано таймер у лівий нижній кут (рис. 4.2) та запущено сцену [23].

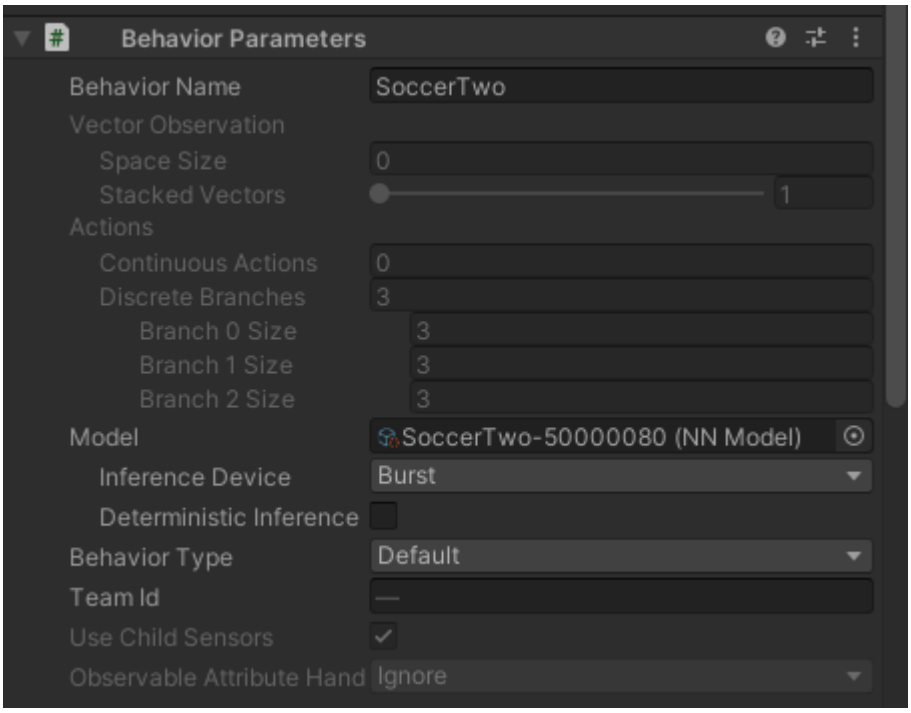

Рисунок 4.1 – Встановлена модель

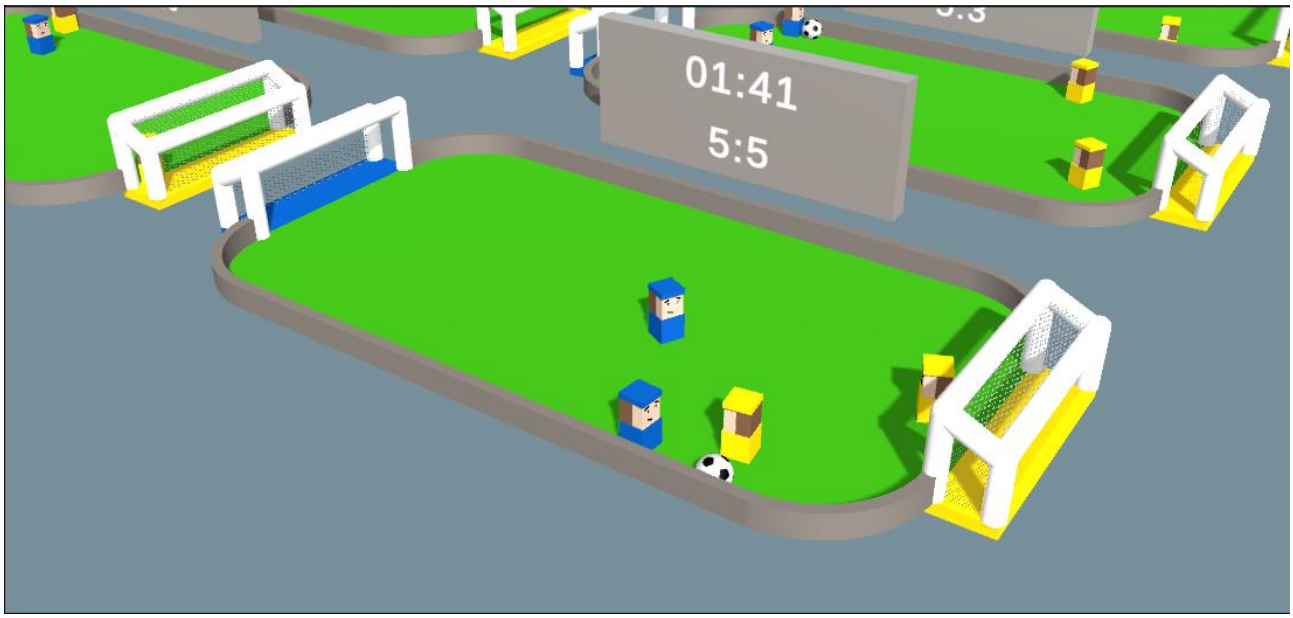

Рисунок 4.2 – Фінальний вигляд запущеної сцени

Для отримання похибки також додано ще 9 середовищ (рис. 4.3), та на основі 10 тестів створено таблицю:

Таблиця 4.1 – Результати тестування

Рахунок	Час (хв)
514 : 486	138
491 : 509	142
552 : 448	139

Продовження таблиці 4.1.

Рахунок	Час (хв)
484 : 516	144
532 : 468	136
499 : 501	145
462 : 538	158
500 : 500	150
513 : 487	139
514 : 486	162

Рисунок 4.3 – Десять тестових середовищ

Отримані результати мають доволі маленьку похибку, а саме середній рахунок за результатами 10 тестів це 506.1 : 493.9 та час 145.3 хв. Це черговий раз підтверджує рівність між гравцями різних команд. Для тих самих параметрів запустимо навчання в іншій конфігурації, а саме 2 гравці (один на один):

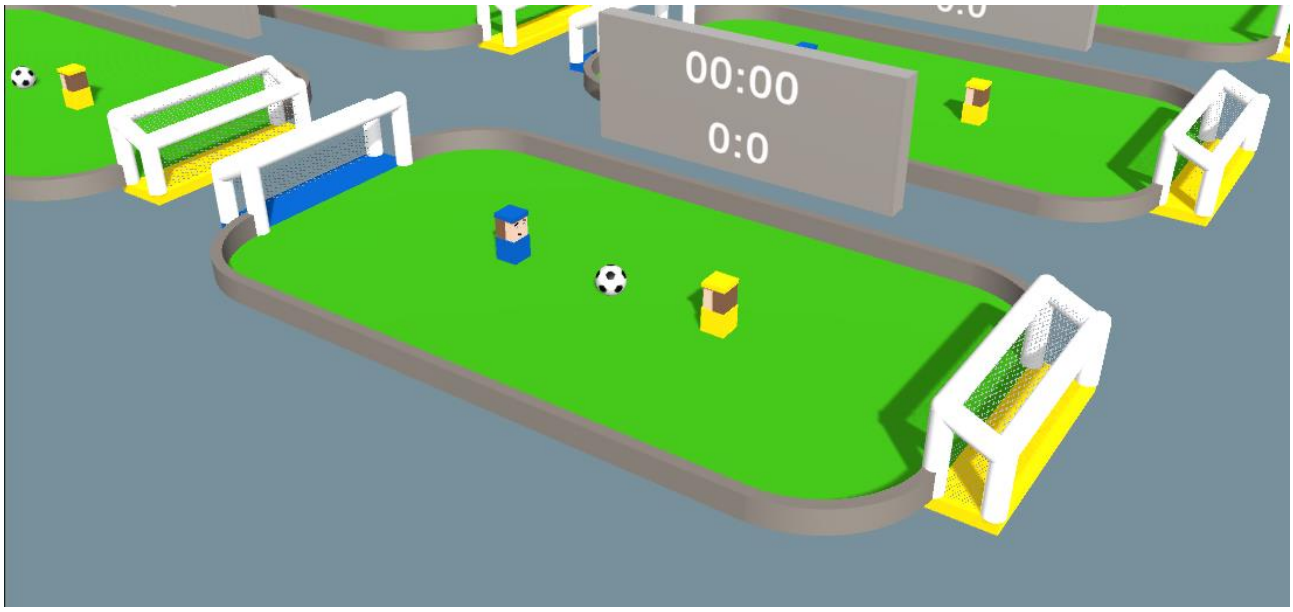

Рисунок 4.4 – Сцена один на один

Таблиця 4.2 – Результати тестування

Рахунок	Час (хв)
524 : 476	135
508 : 492	124
492 : 508	134
515 : 485	142
517 : 483	140
504 : 496	134
481 : 519	132
527 : 473	138
504 : 496	149
509 : 491	128

Отримуємо середні значення 508.1 : 491.9 та час 135.6 хв. Похибка лишилася на тому самому рівні, що і у сцені з чотирма гравцями. при тому витрачена кількість часу не надто сильно відрізняється. Отже можна продовжувати тестування. Після підняття кількості гравців до 6 (рис. 4.5) отримано небажаний результат, а саме нейромережа не змогла навчитися і агенти робили хаотичні рухи навіть через декілька мільйонів ітерацій, зайнявши кути ігрового поля. При цьому вони не чіпали м'яч і забиті голи були лише випадковістю.

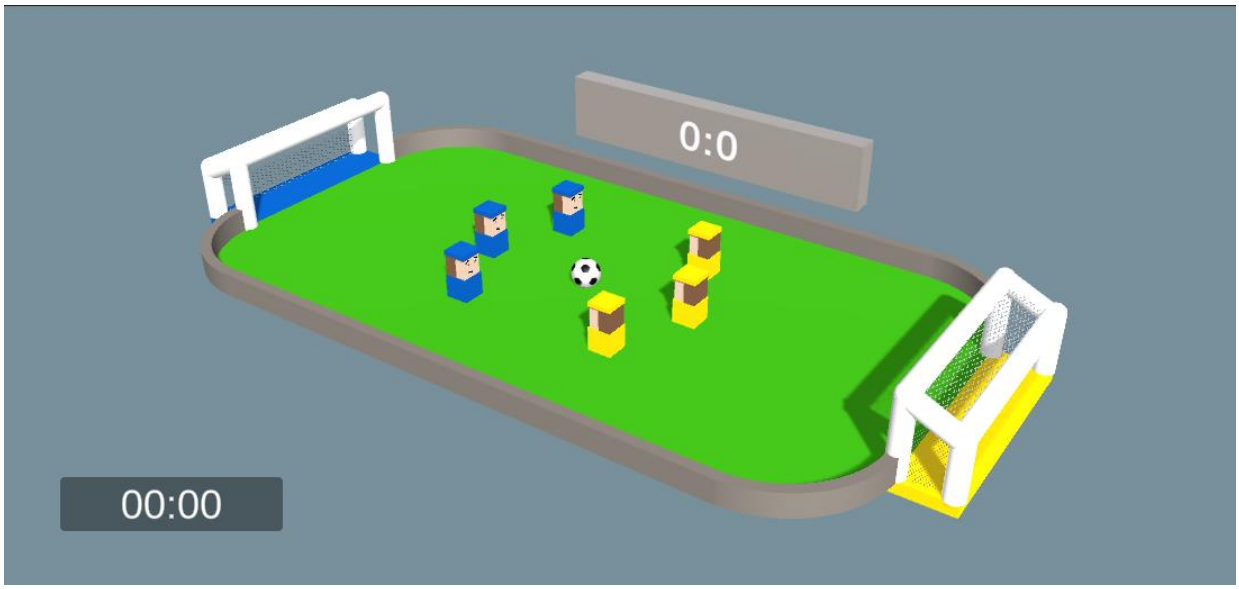

Рисунок 4.5 – Сцена три на три гравці

Після проведення аналізу було прийнято рішення про зміну конфігурації. Було змінена наступні параметри:

– hyperparameters.batch_size з 2048 до 4096;

– hyperparameters.buffer_size з 20480 до 50000;

– hyperparameters.learning_rate з 0.0003 до 0.0002;

– hyperparameters.num_layers з 2 до 3 [24].

Інші параметри залишилися незмінними. Цього разу навчання пройшло успішно. Тепер знову необхідно перевірити якість навчання для конфігурації на 2 та 4 гравці, а також додати сцену на 8 гравців (рис.4.6).

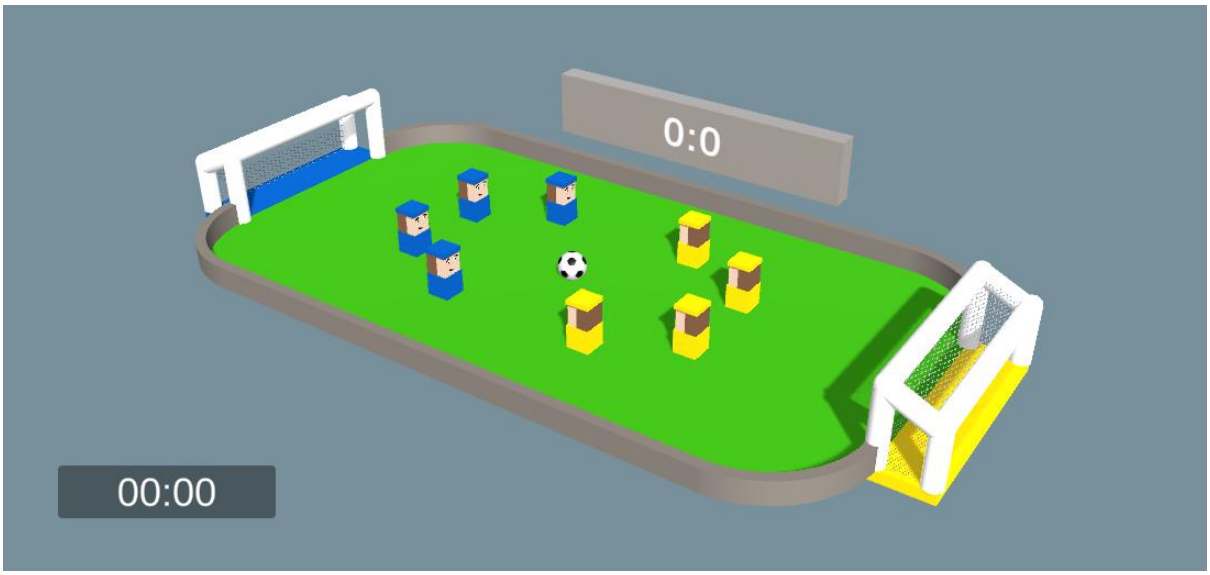

Рисунок 4.6 – Сцена чотири на чотири гравці

Таким чином буде обрано параметри, на яких буде проведено усі тести. Також для створення незвичайних умов додано сцену, яка сильніше за інші відрізняється. Відмінність не у кількості агентів, а у самому вигляді поля в імітаційному середовищі (рис. 4.7). Порівняння цього варіанту зі звичайним полем з такою самою кількістю гравців буде максимально показовим.

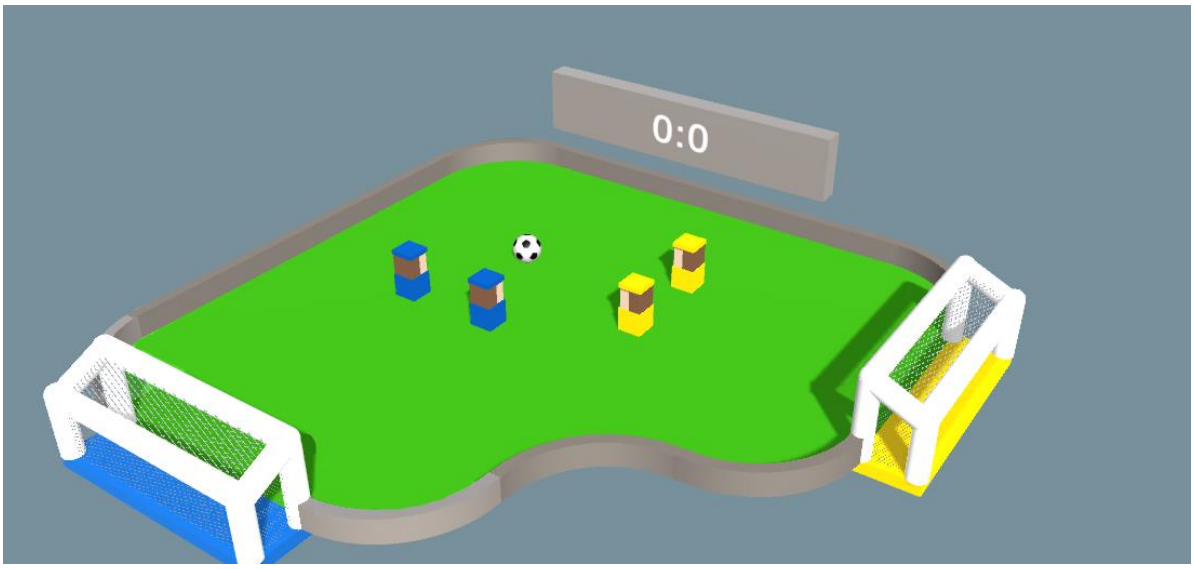

Рисунок 4.7 – Сцена два на два гравці з вигнутим полем

Після проведення тестів результати було додано до табл. 4.3:

Таблиця 4.3 – Результати тестування інших параметрів

Сцена	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
2 гравці, стандартна	Час	157	137	156	150	149	167	162	155	150	154
	Сині	503	552	488	486	503	540	510	502	512	472
	Жовті	497	448	512	514	497	460	490	498	488	528
4 гравці, стандартна	Час	157	155	165	162	171	172	180	175	155	196
	Сині	500	497	532	510	517	482	462	460	505	518
	Жовті	500	503	468	490	483	518	538	540	495	482
6 гравців, стандартна	Час	163	167	167	164	156	173	170	169	178	159
	Сині	477	476	501	497	476	487	508	487	492	503
	Жовті	523	524	499	503	524	513	492	513	508	497
8 гравців, стандартна	Час	198	201	198	196	198	198	196	196	196	206
	Сині	476	533	490	495	504	524	479	492	510	497
	Жовті	524	467	510	505	496	476	521	508	490	503

2 гравці, вигнута	Час	243	228	245	241	242	230	242	250	236	233
	Сині	498	513	517	525	504	484	514	487	515	498
	Жовті	502	487	483	475	496	516	486	513	485	502

Тепер розрахуємо середні значення для кожної сцени за результатами 10 тестів та занесемо до таблиці 4.4.

Таблиця 4.4 – Середні значення за результатами тестування

Сцена	Рахунок	Час (хв)
2 гравці, стандартна	506.8 : 493.2	153.7
4 гравці, стандартна	498.3 : 501.7	168.8
6 гравців, стандартна	490.4 : 509.6	166.6
8 гравців, стандартна	500 : 500	198.3
2 гравці, вигнута	505.5 : 494.5	239

Середнє значення рахунку команд не містить значущої інформації для детального розгляду, оскільки всі значення знаходяться в межах похибки. Однак, цікавою виявилася тривалість ігрового часу. Помітно збільшення часу гри у випадках із 8 гравцями, а також при використанні вигнутого поля.

У випадку вигнутого поля ситуація пояснюється тим, що м'яч часто блокувався у кутах, де 4 гравців було достатньо для його утримання, але для того щоб вибити м'яч, потрібно було більше часу.

Імітаційне середовище з 8 гравцями продемонструвало іншу цікаву поведінку. Тут також спостерігалося блокування м'яча, але змінилася стратегія гри. Нейромережа навчилася ефективніше захищати ворота, і агенти почали будувати бар'єр. Через такий захист кожен гол давався складніше і вимагав більших зусиль. Подібна ситуація прослідковується і у сценах на 4 та 6 гравців, але меншим чином. Таким чином, поки частина команди намагалася заволодіти м'ячем і пробитися до воріт супротивника, інша частина команди ставала на захист.

Такої поведінки не було помічено у сценах з 2 гравцями, що свідчить про значну роль кількості агентів у формуванні стратегії гри. Це спостереження відкриває нові можливості для досліджень. Зокрема, можна спробувати

використовувати моделі зі сцени з 2 гравцями у сцені з 8 гравцями та навпаки. Це дозволить зрозуміти, чи дійсно причина в нестачі гравців для ефективного захисту воріт.

Таким чином, збільшення часу гри у випадках з 4, 6 та 8 гравцями не є похибкою тестів, а свідчить про нові стратегії, які виникають при зміні кількості агентів. Такі результати підкреслюють важливість врахування кількості гравців при розробці імітаційних середовищ для навчання агентів та подальших досліджень у цій галузі.

До перевірки цієї теорії повернемося пізніше, а зараз зосередимося на аналізі швидкості навчання у різних імітаційних середовищах. По черзі буде проведено тести з кроком у 500 тисяч ітерацій до моменту досягнення показників середнього часу гри, які були наведені у таблиці 4.4.

Для обмеження часу було обрано значення у 1000 хвилин, щоб уникнути довгого очікування на завершення гри моделей, які не навчилися ефективно діяти в імітаційному середовищі. Час у 500 хвилин більш ніж вдвічі перевищує максимальний час серед імітаційних середовищ, що було обрано для того, щоб надати достатньо часу для отримання повної картини прогресу у навчанні.

Результати навчання для кожної сцени будуть занесені до окремих таблиць, що дозволить детально проаналізувати процес навчання в кожному середовищі. Цей аналіз є важливим етапом для розуміння того, як різні конфігурації середовищ та кількість агентів впливають на процес навчання.

Таблиця 4.5 – Результати навчання агентів на різній кількості ітерацій для стандартної сцени з 2 гравцями

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
1	Час	500	500	500	500	500	500	500	500	500	500
	Сині	83	72	79	98	80	60	85	96	72	71
	Жовті	73	90	70	77	69	93	78	85	75	55
5	Час	121	112	127	124	120	123	127	123	122	124
	Сині	501	453	479	475	500	478	502	456	517	487

	Жовті	499	547	521	525	500	522	498	544	483	513
10	Час	124	119	123	130	115	124	126	130	126	128
	Сині	518	464	483	485	492	496	523	507	486	498
	Жовті	482	536	517	515	508	504	477	493	514	502

Продовження таблиці 4.5.

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
15	Час	124	114	122	128	127	123	127	122	126	129
	Сині	511	467	492	503	497	534	540	539	501	507
	Жовті	489	533	508	497	503	466	460	461	499	493
20	Час	123	142	139	132	137	148	133	141	139	132
	Сині	490	483	508	495	534	539	517	530	513	506
	Жовті	510	517	492	505	466	461	483	470	487	494
25	Час	152	138	149	149	144	167	144	139	144	149
	Сині	497	501	534	536	513	525	499	501	495	508
	Жовті	503	499	466	464	487	475	501	499	505	492
30	Час	133	123	136	128	130	140	135	132	133	12
	Сині	522	475	531	521	488	495	523	526	497	505
	Жовті	478	525	469	479	512	505	477	474	503	495
35	Час	136	124	138	133	131	136	133	128	142	138
	Сині	511	484	520	485	482	478	508	508	490	492
	Жовті	489	516	480	515	518	522	492	492	510	508
40	Час	139	137	145	139	141	147	150	148	156	141
	Сині	502	491	505	491	486	487	525	515	533	505
	Жовті	498	509	495	509	514	513	475	485	467	495
45	Час	148	138	155	146	146	152	144	151	150	139
	Сині	520	512	505	532	512	503	491	465	525	497
	Жовті	480	488	495	468	488	497	509	535	475	503

Таблиця 4.6 – Середні результати навчання агентів на різній кількості ітерацій для стандартної сцени з 2 гравцями

Кількість ітерацій (млн)	Рахунок	Час (хв)
1	79.6 : 76.5	500
5	484.8 : 515.2	122.3
10	495.2 : 504.8	124.5
15	509.1 : 490.9	124.2
20	511.5 : 488.5	136.6
25	510.9 : 489.1	147.5

30	508.3 : 491.7	120.2
35	495.8 : 504.2	133.9
40	504 : 496	144.3
45	506.2 : 493.8	146.9

Аналізуючи результати, можна побачити, що нейромережі знадобилося не більше 5 мільйонів кроків, аби навчитися поводитися в імітаційному середовщі. Але чи усі сцени матимуть схожий результат? Порівняємо з прикладом на 4 гравці:

Таблиця 4.7 – Результати навчання агентів на різній кількості ітерацій для стандартної сцени з 4 гравцями

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
1	Час	500	500	500	500	500	500	500	500	500	500
	Сині	20	24	18	17	18	13	17	26	16	16
	Жовті	28	24	20	16	19	18	15	20	12	13
5	Час	500	500	500	500	500	500	500	500	500	500
	Сині	170	200	198	120	180	156	140	180	188	140
	Жовті	158	188	182	117	175	145	135	172	177	125
10	Час	126	115	130	128	129	128	144	138	131	162
	Сині	519	484	517	500	505	489	438	481	522	508
	Жовті	481	517	483	500	495	511	562	519	478	492
15	Час	111	115	121	122	110	122	132	129	112	142
	Сині	505	481	532	523	532	524	429	492	524	510
	Жовті	495	519	468	477	468	476	571	508	476	490
20	Час	119	120	125	131	123	131	138	133	130	146
	Сині	489	491	523	517	530	472	440	491	552	503
	Жовті	511	509	477	483	470	528	560	509	449	497
25	Час	125	127	128	126	131	129	130	123	143	152
	Сині	483	487	552	490	513	454	504	525	451	501
	Жовті	517	513	448	510	487	546	496	475	549	499
30	Час	147	135	141	143	140	148	155	154	141	162
	Сині	502	514	526	525	530	444	445	473	527	493
	Жовті	498	486	474	475	470	556	555	527	473	507
35	Час	158	160	171	174	161	162	182	180	162	213
	Сині	514	549	547	523	535	465	450	483	521	498
	Жовті	486	451	453	477	465	535	550	517	479	502

40	Час	155	165	165	166	164	173	193	190	162	199
	Сині	500	495	547	509	530	477	476	467	530	481
	Жовті	500	505	453	491	470	523	524	533	470	519
45	Час	153	151	155	160	155	161	175	170	158	174
	Сині	492	505	508	506	503	458	440	488	520	495
	Жовті	508	495	492	494	497	542	560	512	480	505

Таблиця 4.8 – Середні результати навчання агентів на різній кількості ітерацій для стандартної сцени з 4 гравцями

Кількість ітерацій (млн)	Рахунок	Час (хв)
1	18.5 : 18.5	500
5	167.2 : 157.4	500
10	496.3 : 503.8	133.1
15	505.2 : 494.8	121.6
20	500.8 : 499.3	129.6
25	496 : 504	131.4
30	497.9 : 502.1	146.6
35	508.5 : 491.5	172.3
40	501.2 : 498.8	173.2
45	491.5 : 508.5	161.2

Цього разу нейромережі було явно недостатньо 5 мільйонів ітерацій для досягнення значущих результатів, і лише після 10 мільйонів ітерацій почали з'являтися помітні зміни. Варто зазначити ще одну невелику відмінність: час гри також відрізняється в обох випадках. На початку навчання час гри був меншим, але згодом він збільшувався.

При детальному аналізі процесу навчання стає помітно, що нейромережа поступово вдосконалювала свої навички захисту воріт. Агенти все краще і краще захищали свої ворота від м'яча, що призводило до збільшення часу гри. Вони регулярно повертались назад та ставали до воріт, блокуючи потрапляння м'яча.

Тепер виконаємо тестування нейромереж, що залишилися.

Таблиця 4.9 – Результати навчання агентів на різній кількості ітерацій для стандартної сцени з 6 гравцями

	Опис	Результат

Кількість ітерацій (млн)		1	2	3	4	5	6	7	8	9	10
1	Час	500	500	500	500	500	500	500	500	500	500
	Сині	16	22	22	19	28	22	22	14	20	14
	Жовті	21	26	20	24	24	19	20	15	24	14
5	Час	217	211	222	230	217	229	237	227	224	214
	Сині	496	510	487	491	499	497	505	505	502	486
	Жовті	504	490	513	509	501	503	495	495	498	514

Продовження таблиці 4.9

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
10	Час	194	173	178	193	187	191	177	193	195	186
	Сині	501	483	507	510	509	502	517	502	502	498
	Жовті	499	517	493	490	491	498	483	498	498	502
15	Час	174	160	172	173	173	178	170	167	170	178
	Сині	493	523	494	499	493	489	504	502	480	490
	Жовті	507	477	506	501	507	511	496	498	520	510
20	Час	143	139	141	145	147	158	144	156	146	157
	Сині	508	514	515	509	511	506	495	505	486	491
	Жовті	492	486	485	491	489	494	505	495	514	509
25	Час	150	135	147	145	146	145	150	147	137	147
	Сині	518	492	486	529	479	515	473	493	491	505
	Жовті	482	508	514	471	521	485	527	507	509	495
30	Час	166	155	171	172	168	168	171	159	175	161
	Сині	519	496	530	482	502	494	487	486	502	508
	Жовті	481	504	470	518	498	506	513	514	498	492
35	Час	157	162	160	159	162	157	158	161	154	154
	Сині	502	490	484	498	537	514	513	503	496	514
	Жовті	498	510	516	502	463	486	487	497	504	486
40	Час	167	154	171	163	166	168	157	167	172	167
	Сині	491	489	526	500	490	531	506	509	513	488
	Жовті	509	511	474	500	510	469	494	491	487	512
45	Час	175	163	173	170	165	172	167	170	179	168
	Сині	518	533	485	460	495	503	500	525	513	460
	Жовті	482	467	515	540	505	497	500	475	487	540

Таблиця 4.10 – Середні результати навчання агентів на різній кількості ітерацій для стандартної сцени з 6 гравцями

Кількість ітерацій (млн)	Рахунок	Час (хв)
1	19.9 : 20.7	500
5	497.8 : 502.2	222.8
10	503.1 : 496.9	186.7
15	496.7 : 503.3	171.5
20	504 : 496	147.6
25	498.1 : 501.9	144.9
30	500.6 : 499.4	166.6
35	505.1 : 494.9	158.4

Продовження таблиці 4.10.

Кількість ітерацій (млн)	Рахунок	Час (хв)
40	504.3 : 495.7	165.2
45	499.2 : 500.8	170.2

Таблиця 4.11 – Результати навчання агентів на різній кількості ітерацій для стандартної сцени з 8 гравцями

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
1	Час	500	500	500	500	500	500	500	500	500	500
	Сині	58	56	60	55	68	50	67	63	52	65
	Жовті	63	66	51	58	70	65	57	54	65	63
5	Час	314	293	290	302	281	292	304	291	296	303
	Сині	499	516	494	504	477	517	516	497	483	525
	Жовті	501	484	506	496	523	483	484	503	517	475
10	Час	225	214	207	214	216	223	222	223	219	225
	Сині	521	472	486	464	514	525	508	499	521	517
	Жовті	479	528	514	536	486	475	492	501	479	483
15	Час	160	157	156	149	156	154	156	156	166	158
	Сині	500	505	508	510	520	481	486	512	474	500
	Жовті	500	495	492	490	480	519	514	488	526	500
20	Час	186	182	191	188	185	179	178	193	191	200
	Сині	500	501	523	508	490	496	502	474	524	484
	Жовті	500	499	477	492	510	504	498	526	476	516
25	Час	224	230	234	224	235	235	233	241	235	233
	Сині	489	491	513	471	500	501	500	478	501	492
	Жовті	511	509	487	529	500	499	500	522	499	508
30	Час	180	182	180	180	180	186	187	176	179	182

	Сині	500	491	507	470	461	509	508	513	503	466
	Жовті	500	509	493	530	539	491	492	487	497	534
35	Час	194	182	186	190	195	197	189	196	182	191
	Сині	498	495	495	513	501	488	470	484	502	497
	Жовті	502	505	505	487	499	512	530	516	498	503
40	Час	169	169	179	180	175	176	174	170	164	171
	Сині	514	520	504	480	517	483	538	482	495	521
	Жовті	486	480	496	520	483	517	462	518	505	479
45	Час	194	191	190	191	193	188	178	189	184	183
	Сині	469	495	489	498	490	487	492	475	477	483
	Жовті	531	505	511	502	510	513	508	525	523	517

Таблиця 4.12 – Середні результати навчання агентів на різній кількості ітерацій для стандартної сцени з 8 гравцями

Кількість ітерацій (млн)	Рахунок	Час (хв)
1	59.4 : 61.2	500
5	502.8 : 497.2	296.6
10	502.7 : 497.3	218.8
15	499.6 : 500.4	156.8
20	500.2 : 499.8	187.3
25	493.6 : 506.4	232.4
30	492.8 : 507.2	181.2
35	494.3 : 505.7	190.2
40	505.4 : 494.6	172.7
45	485.5 : 514.5	188.1

Таблиця 4.13 – Результати навчання агентів на різній кількості ітерацій для вигнутої сцени з 4 гравцями

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
1	Час	500	500	500	500	500	500	500	500	500	500
	Сині	23	25	28	29	23	25	21	20	22	35
	Жовті	17	13	20	25	20	23	10	16	21	14
5	Час	254	244	261	263	251	251	251	237	260	250
	Сині	539	529	530	537	514	522	525	547	501	480
	Жовті	461	471	470	463	486	478	475	453	499	520
10	Час	179	196	186	194	187	190	196	202	181	186
	Сині	497	431	465	532	572	440	538	473	491	454

	Жовті	503	569	535	468	428	560	462	527	509	546
15	Час	179	166	176	182	176	187	185	172	178	177
	Сині	414	438	567	572	572	529	418	429	434	550
	Жовті	586	562	433	428	428	571	582	571	566	450
20	Час	259	238	240	247	256	237	262	266	254	228
	Сині	451	475	496	540	454	560	497	487	462	525
	Жовті	549	525	504	460	546	440	503	513	538	475
25	Час	185	184	201	186	177	180	189	188	185	205
	Сині	457	511	476	533	465	527	467	534	473	474
	Жовті	543	489	524	467	535	473	533	466	527	526

Продовження таблиці 4.13

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
30	Час	257	256	267	263	263	276	278	300	251	269
	Сині	431	540	455	464	449	560	572	438	525	550
	Жовті	569	460	545	536	551	440	428	562	475	450
35	Час	198	185	211	204	193	200	200	203	185	198
	Сині	528	561	522	448	479	483	545	516	518	514
	Жовті	472	439	478	552	521	517	455	484	482	486
40	Час	234	229	233	223	221	224	252	252	226	239
	Сині	504	533	450	520	487	502	517	452	535	480
	Жовті	496	467	550	480	513	498	483	548	465	520
45	Час	218	205	219	202	219	220	237	231	211	210
	Сині	502	530	520	499	549	497	518	510	535	478
	Жовті	498	470	480	501	451	503	482	490	465	522

Таблиця 4.14 – Середні результати навчання агентів на різній кількості ітерацій для вигнутої сцени з 4 гравцями

Кількість ітерацій (млн)	Рахунок	Час (хв)
1	25.1 : 17.9	500
5	522.4 : 477.6	252.2
10	489.3 : 510.7	189.7
15	492.3 : 517.7	177.8
20	494.7 : 505.3	248.7
25	491.7 : 508.3	188
30	498.4 : 501.6	268

35	511.4 : 488.6	197.7
40	498 : 502	233.3
45	513.8 : 486.2	217.2

На основі цих результатів можна зробити висновок, що в більшості випадків нейромережі для гри достатньо 5 мільйонів ітерацій. Лише для стандартної сцени з 4 гравцями знадобилося більше кроків – 10 мільйонів. В інших випадках результати збігалися, і навчання проходило однаково. Спочатку час тестування зменшувався, і приблизно на середині навчання починав підійматися.

Тепер необхідно перевірити, чи було виправдано чекати усі 50 мільйонів ітерацій. Для цього буде порівняно результати на кількості ітерацій, коли нейромережа навчилася нормально грати. Це будуть раніше згадані 5 або 10 мільйонів кроків. Також буде проведено порівняння різниці у навчанні на різних етапах. Для випробувань обрано найбільш показові комбінації:

– 5 або 10 мільйонів ітерацій (залежно від випадку) проти 50 мільйонів;

– 5 або 10 мільйонів ітерацій (залежно від випадку) проти наступного ступеню (10 або 15 мільйонів);

– 45 мільйонів ітерацій проти 50 мільйонів ітерацій.

Ці порівняння дозволять оцінити ефективність додаткових ітерацій у навчанні нейромережі. Зокрема, ми зможемо визначити, чи дійсно значне збільшення кількості ітерацій (до 50 мільйонів) призводить до суттєвого покращення результатів, чи навчання стабілізується вже на більш ранніх етапах.

Проведені тести допоможуть виявити оптимальну кількість ітерацій для ефективного навчання агентів у різних середовищах. Це дозволить не лише зекономити час та ресурси, але й оптимізувати процес розробки імітаційних середовищ для майбутніх досліджень та застосувань [25].

Тож, цього разу агенти, що належать до синьої команди, будуть виступати в якості еталону нейромережі та матимуть більшу кількість кроків. Жовта команда матиме кількість кроків залежно від поточної комбінації.

Таблиця 4.15 – Тестування комбінацій ітерацій для стандартної сцени на 2 гравці

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
5 проти 50	Час	129	129	120	128	125	126	123	124	121	120
	Сині	724	748	717	719	715	736	717	737	709	737
	Жовті	276	252	283	281	285	264	283	263	291	263
5 проти 10	Час	121	119	120	119	118	122	120	119	127	125
	Сині	681	656	640	676	674	667	650	656	622	663
	Жовті	319	344	360	324	326	333	350	344	378	337

Продовження таблиці 4.15

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
45 проти 50	Час	141	144	157	145	143	148	155	157	154	139
	Сині	531	522	542	564	511	511	491	527	521	508
	Жовті	469	478	458	436	489	489	509	473	479	492

Таблиця 4.16 – Середні результати тестування комбінацій ітерацій для стандартної сцени на 2 гравці

Кількість ітерацій (млн)	Рахунок	Час (хв)
5 проти 50	725.9 : 274.1	124.5
5 проти 10	658.5 : 341.5	121
45 проти 50	522.8 : 477.2	148.3

Таблиця 4.17 – Тестування комбінацій ітерацій для стандартної сцени на 4 гравці

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
10 проти 50	Час	79	87	78	87	83	81	90	87	85	86
	Сині	977	975	980	982	989	979	979	975	978	984
	Жовті	23	25	20	18	11	21	21	25	22	16
10 проти 15	Час	90	90	92	92	91	99	96	98	93	97
	Сині	959	964	965	970	965	973	976	971	979	972
	Жовті	41	36	35	30	35	27	24	29	21	28
	Час	152	156	159	160	154	162	175	174	165	192

45 проти 50	Сині	495	481	503	501	531	458	443	485	520	502
	Жовті	505	519	497	499	469	542	557	515	480	498

Таблиця 4.18 – Середні результати тестування комбінацій ітерацій для стандартної сцени на 4 гравці

Кількість ітерацій (млн)	Рахунок	Час (хв)
10 проти 50	725.9 : 274.1	124.5
10 проти 15	658.5 : 341.5	121
45 проти 50	522.8 : 477.2	148.3

Таблиця 4.19 – Тестування комбінацій ітерацій для стандартної сцени на 6 гравців

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
5 проти 50	Час	114	118	118	117	115	113	115	110	111	111
	Сині	919	902	898	917	916	899	918	918	893	917
	Жовті	81	98	102	83	84	101	82	85	107	83
5 проти 10	Час	138	137	141	137	140	134	138	129	144	139
	Сині	882	876	883	893	888	883	892	873	878	885
	Жовті	118	124	117	107	112	117	108	127	122	115
45 проти 50	Час	170	170	171	165	164	170	161	162	175	164
	Сині	481	504	476	504	490	484	489	492	476	488
	Жовті	519	497	524	496	510	516	511	508	524	512

Таблиця 4.20 – Середні результати тестування комбінацій ітерацій для стандартної сцени на 6 гравців

Кількість ітерацій (млн)	Рахунок	Час (хв)
5 проти 50	909.7 : 90.6	114.2
5 проти 10	883.3 : 116.7	137.7
45 проти 50	488.4 : 511.6	167.2

Таблиця 4.21 – Тестування комбінацій ітерацій для стандартної сцени на 8 гравців

	Опис	Результат

Кількість ітерацій (млн)		1	2	3	4	5	6	7	8	9	10
5 проти 50	Час	146	145	189	139	139	142	151	138	143	146
	Сині	932	932	934	934	926	911	926	935	919	932
	Жовті	68	68	66	66	74	89	74	65	81	68
5 проти 10	Час	171	168	171	177	173	164	177	173	175	181
	Сині	905	909	904	890	902	918	892	888	881	883
	Жовті	95	91	96	110	98	82	108	112	119	117
45 проти 50	Час	195	197	178	182	203	198	188	193	190	188
	Сині	485	473	499	515	500	502	468	471	486	494
	Жовті	515	527	501	485	500	498	532	529	514	506

Таблиця 4.22 – Середні результати тестування комбінацій ітерацій для стандартної сцени на 8 гравців

Кількість ітерацій (млн)	Рахунок	Час (хв)
5 проти 50	928.1 : 71.9	147.8
5 проти 10	897.2 : 102.8	173
45 проти 50	489.3 : 510.7	191.2

Таблиця 4.23 – Тестування комбінацій ітерацій для вигнутої сцени на 4 гравці

Кількість ітерацій (млн)	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
5 проти 50	Час	165	155	157	161	166	163	158	172	155	158
	Сині	901	909	903	887	888	877	898	894	891	896
	Жовті	99	91	97	113	112	123	102	106	109	104
5 проти 10	Час	180	166	190	190	177	177	183	173	176	177
	Сині	801	784	817	819	793	794	807	784	801	792
	Жовті	199	216	183	181	207	206	193	216	199	208
45 проти 50	Час	246	240	247	247	232	248	229	237	234	264
	Сині	520	528	519	539	524	494	525	518	481	520
	Жовті	480	472	481	461	476	506	475	482	519	481

Таблиця 4.24 – Середні результати тестування комбінацій ітерацій для вигнутої сцени на 4 гравці

Кількість ітерацій (млн)	Рахунок	Час (хв)
5 проти 50	894.4 : 105.6	161
5 проти 10	799.2 : 200.8	178.9
45 проти 50	516.8 : 483.3	242.4

Помітно, що кількість ітерацій, на якій нейромережа навчилася нормально грати, є ще досить низьким показником порівняно із наступним кроком навчання. Різниця у 5 мільйонів ітерацій виявилася значною в усіх випадках для всіх імітаційних середовищ. Це свідчить про те, що додаткові ітерації суттєво впливають на покращення результатів.

На цьому етапі ми вже впевнилися, що всі нейромережі мають дуже схожий процес еволюції та окремо одна від одної пройшли однаковий шлях навчання. Така схожість підтверджує стабільність і передбачуваність процесу навчання для різних моделей.

З огляду на вищезгадані моменти, порівняння нейромереж з різних імітаційних середовищ буде максимально показовим. Воно дозволить відобразити залежність якості навчання від створених умов та допоможе виявити ключові фактори, що впливають на ефективність навчання. Це порівняння також допоможе визначити оптимальні налаштування для створення імітаційних середовищ, що сприятиме підвищенню якості навчання агентів у майбутніх дослідженнях.

Результати цього порівняння матимуть важливе значення для подальшого розвитку моделей навчання з підкріпленням, оскільки вони дадуть змогу більш точно налаштовувати параметри імітаційних середовищ, забезпечуючи таким чином ефективніше навчання агентів.

Як і у попередній раз, еталонна модель навчання буде притаманна синій команді, а жовта команда буде змінювати мозок для порівняння. Отже, для синьої команди встановлено мозок з 50 мільйонами ітерацій, що пройшли навчання на стандартній сцені з 4 гравцями. Результати тестування занесемо до таблиці:

Таблиця 4.25 – Тестування різних нейромереж на стандартній сцені з 4 гравцями

Опис нейромережі жовтої команди	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
2 гравці, стандартна сцена	Час	136	125	128	132	139	139	145	145	138	162
	Сині	450	489	492	449	492	438	393	475	474	458
	Жовті	550	511	508	551	508	562	607	525	526	542
4 гравці, вигнута сцена	Час	149	147	167	159	154	176	181	182	160	180
	Сині	726	735	761	745	763	763	726	769	762	770
	Жовті	274	265	239	255	237	237	274	231	238	230
6 гравців, стандартна сцена	Час	130	130	134	132	136	138	144	145	137	158
	Сині	650	666	687	636	683	661	620	638	700	704
	Жовті	350	334	313	364	317	339	380	362	300	296

Продовження таблиці 4.25

Опис нейромережі жовтої команди	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
8 гравців, стандартна сцена	Час	119	119	122	120	129	132	134	125	138	142
	Сині	773	773	797	788	739	772	752	789	734	798
	Жовті	227	227	203	212	261	228	248	211	266	202

Таблиця 4.26 – Середні результати тестування різних нейромереж на стандартній сцені з 4 гравцями

Кількість ітерацій (млн)	Рахунок	Час (хв)
2 гравці, стандартна сцена	461 : 539	138.9
4 гравці, вигнута сцена	752 : 248	165.5
6 гравців, стандартна сцена	664.5 : 335.5	138.4
8 гравців, стандартна сцена	771.5 : 228.5	128

Отримуємо неочікуваний результат у першому тесті. Нейромережа, яка навчалася у комбінації 1 проти 1 агента, виявилася кращою за результатами всіх тестів проти нейромережі, що навчалася у тестовій комбінації. При повторному

перегляді геймплею стає зрозумілим, чому так сталося. Агенти, що навчалися без команди, грають так само у команді. Вони нічого не знають про другого гравця і виконують дії синхронно: двоє намагаються забити м'яча або двоє намагаються захистити ворота.

В інших випадках результати були більш очікуваними: нейромережа, навчена у тестовій комбінації, виявилася кращою. Різниця у продуктивності між нейромережами була також значно більшою, ніж очікувалося.

Наступний крок полягає у перевірці здатності нейромереж адаптуватися до нових умов. Для цього помістимо нейромережі у сцену, в якій жодна з них раніше не була. Нова сцена міститиме 4 гравців, але на полі будуть встановлені додаткові бар'єри (рис. 4.8). Це дозволить побачити, чи збережуться результати навчання в умовах зміненого середовища.

Цей підхід допоможе визначити, наскільки ефективно нейромережі можуть адаптувати свої стратегії до нових умов гри та як додаткові перешкоди вплинуть на їхню продуктивність. Результати цього тесту нададуть цінну інформацію про гнучкість та універсальність розроблених моделей, що є важливим для їх подальшого вдосконалення і застосування у різних сценаріях.

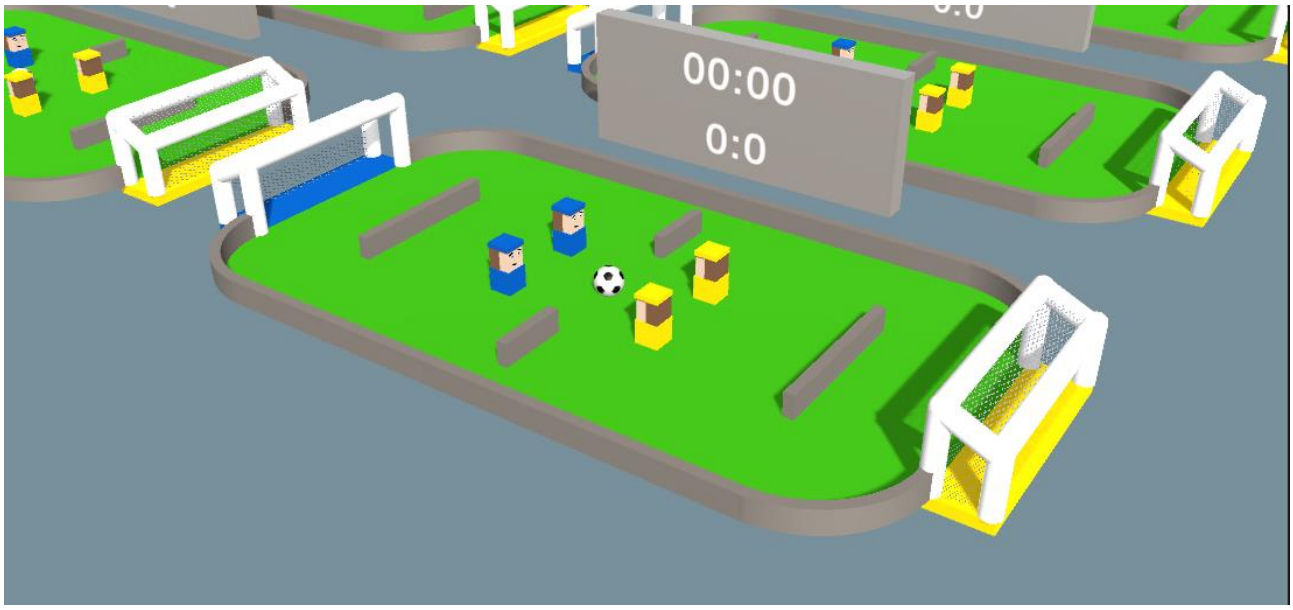

Рисунок 4.8 – Поле з додатковими бар'єрами

Таблиця 4.27 – Тестування різних нейромереж на сцені з бар'єрами

Опис нейромережі команди	Опис	Результат									
		1	2	3	4	5	6	7	8	9	10
2 гравці, стандартна сцена, 50млн	Час	206	209	219	216	214	204	212	215	218	210
	Сині	503	495	503	488	517	486	514	515	519	497
	Жовті	497	505	497	512	483	514	486	485	481	503
4 гравці, стандартна сцена, 50млн	Час	198	189	192	198	186	201	189	194	189	191
	Сині	488	521	494	517	501	470	477	505	496	515
	Жовті	512	479	506	483	499	530	523	495	504	485
Комбінація 1	Час	200	205	205	207	210	210	216	218	227	233
	Сині	499	509	497	500	503	479	489	489	529	527
	Жовті	501	491	503	500	497	521	511	511	471	473
Комбінація 2	Час	232	240	231	227	225	238	225	223	226	211
	Сині	460	464	459	480	487	471	481	514	463	486
	Жовті	540	536	541	520	513	529	519	486	537	514

Таблиця 4.28 – Тестування різних нейромереж на сцені з бар'єрами

Кількість ітерацій (млн)	Рахунок	Час (хв)
2 гравці, стандартна сцена, 50млн	503.7 : 496.3	212.3
4 гравці, стандартна сцена, 50млн	498.4 : 501.6	192.7
Комбінація 1	502.1 : 497.9	213.1
Комбінація 2	476.5 : 523.5	227.8

У перших двох випадках було отримано лише середні значення часу гри. Рахунок був доволі рівним і змінювався на користь тієї чи іншої команди. У третьому тесті результати були аналогічні, але в якості першої комбінації гравців для синьої команди було обрано нейромережу, що пройшла навчання на стандартній сцені з 4 гравцями, а для жовтої команди – нейромережу, що навчалася на стандартній сцені з 2 гравцями.

На цьому етапі помітна відмінність від тестування на стандартній сцені. На стандартній сцені нейромережа зі сцени з 2 гравцями перемогла нейромережу зі сцени з 4 гравцями. Чому цього разу результати відрізняються? При перегляді

гри можна помітити, що цього разу нейромережі вели себе досить схоже через додаткові бар'єри, які заважали агентам визначити, чи захищає інший агент ворота.

Остання комбінація містила агентів зі сцени з 6 гравцями (жовта команда) та зі сцени з 8 гравцями (синя команда). Тут спостерігалася перевага синьої команди у більшості тестів. Особливостей поведінки не було помічено.

Ця сцена показала ще одну відмінність від інших. Незважаючи на незначне збільшення часу гри, агенти досить часто втрачали м'яч за бар'єрами і не знали, що робити, застрягаючи на місці. Це свідчить про те, що додаткові перешкоди можуть значно впливати на стратегію гри та поведінку агентів.

Таким чином, результати цього тестування вказують на важливість врахування різних факторів середовища під час навчання нейромереж. Додаткові бар'єри змушують агентів адаптувати свої стратегії, що може вплинути на загальну ефективність їхньої гри. Подальші дослідження повинні зосередитися на вивченні цих факторів та їх впливу на процес навчання і поведінку агентів у різних умовах.

В останньому тесті запропоновано навчання нейромережі проти вже навченої. Узято еталонну сцену на 4 гравці, гравцям синьої команди надано нейромережу з 50 мільйонами ітерацій для цієї сцени, а жовта команда почала навчання за самого початку:

Таблиця 4.29 – Тестування навчання нової нейромережі проти навченої

Опис	Результат									
	1	2	3	4	5	6	7	8	9	10
Час	168	178	174	173	172	174	178	180	164	207
Сині	520	532	537	540	524	488	471	548	565	512
Жовті	480	468	463	460	476	512	529	452	435	488

В середньому, значення рахунку не надто сильно відрізняються, з показником 523.7 проти 476.3 на користь синіх. Були випадки, коли жовті перемагали і зі значною різницею в рахунку. Такі результати можна пояснити складністю навчання на перших етапах – навчені агенти синьої команди краще

знали, як діяти, і не давали нейромережі жовтої команди встигати навчатися в імітаційному середовищі.

Це свідчить про те, що початкові умови навчання і стартові стратегії можуть мати значний вплив на результати гри. Агенти, які мали перевагу в знаннях та стратегіях на початкових етапах, продовжували демонструвати кращі результати протягом усієї гри. Це також може вказувати на те, що процес навчання потребує не лише тривалого часу, але й адаптації до різних умов та стратегії гри суперника.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було здійснено комплексний аналіз існуючих технологій і методик у сфері штучного інтелекту, з особливим акцентом на навчанні з підкріпленням і використанні імітаційних середовищ. Дослідження було направлено на вирішення актуальних задач, пов'язаних з оптимізацією процесів тренування нейронних мереж, а також розробкою та адаптацією імітаційних сценаріїв, які максимально точно відтворюють реальні умови взаємодії агентів.

Основна увага під час досліджень була зосереджена на визначенні та аналізі ефективності різних стратегій навчання, які дозволяють агентам самостійно вчитися на власних помилках без безпосередньої взаємодії з людиною. Було розроблено план для створення моделей імітаційних середовищ, проведення їх тестування та оцінено вплив різних параметрів на ефективність навчання.

Паралельно з технічними аспектами, значна увага була приділена аналізу теоретичних основ штучного інтелекту та машинного навчання. Це дозволило не лише покращити розуміння процесів, що відбуваються під час тренування агентів, але й сформувати міцну основу для подальших досліджень у цій області.

Дослідження виявило, що застосування розроблених методів і підходів може суттєво підвищити якість та швидкість навчання нейронних мереж, забезпечуючи їх здатність до самостійного вирішення складних завдань. Результати практики свідчать про високий потенціал застосування імітаційних середовищ для тренування штучних інтелектів, зокрема в ігровій індустрії, автомобільній промисловості, робототехніці та інших сферах.

Варто зазначити, що попри позитивні аспекти, в ході дослідження також були ідентифіковані певні виклики та обмеження. Серед них – необхідність значних обчислювальних ресурсів для проведення тренувань нейронних мереж і високі витрати часу на адаптацію та налагодження імітаційних середовищ, а також запропоновано варіанти вирішення цих проблем.

Незважаючи на виявлені труднощі, отримані результати та здобуті знання під час проходження науково-дослідної практики дали змогу глибше зрозуміти поточний стан та перспективи розвитку області штучного інтелекту та машинного навчання, зокрема у сфері навчання з підкріпленням. Отримані під час практики знання та досвід будуть використані для подальшого розвитку наукових і практичних навичок, а також для впровадження інноваційних рішень у майбутніх дослідницьких та розробницьких проектах.

Аналіз результатів, отриманих у 4 розділі, показав, що для більшості сценаріїв навчання агентів достатньо 5 мільйонів ітерацій, що дозволяє суттєво скоротити час та ресурси, необхідні для тренування нейронних мереж. Однак для складніших середовищ, як-от сцена з 4 гравцями, необхідно до 10 мільйонів ітерацій. Навчанних на малій кількості ітерацій агентів також можна застосувати або для рівнів складності у грі, або як персонажів, які мають роботи не надто очевидні кроки, але досягають результату.

Вивчення різних конфігурацій середовищ (2, 4, 6 та 8 гравців) показало, що кількість агентів суттєво впливає на стратегії навчання та тривалість ігрових сценаріїв. Збільшення кількості агентів призводить до складніших взаємодій та вимагає більшої кількості ітерацій для досягнення оптимальних результатів. Додавання бар'єрів на полі значно змінює стратегії агентів, що демонструє адаптивність нейронних мереж до нових умов. Це підтверджує необхідність регулярного налаштування параметрів середовища для забезпечення ефективного навчання. Результати навчання агентів показали стабільність незалежно від конфігурації середовища, що підтверджує надійність обраного підходу до навчання з підкріпленням. При цьому, максимальна схожість імітаційного середовища на цільове є запорукою максимального успіху.

Ці висновки підкреслюють важливість комплексного підходу до розробки імітаційних середовищ та алгоритмів навчання з підкріпленням, що забезпечує високу адаптивність та ефективність моделей у різних умовах застосування.

Результати роботи з імітаційними середовищами опубліковано у журналі, що індексується у наукометричній базі Scopus [26].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Modelling and Simulation in the Science of Micro- and Meso-Porous Materials. C. Richard A. Catlow, Veronique Van Speybroeck and Rutger A. van Santen. 2017. 370c.

2. Simulation and Modeling Methodologies, Technologies and Applications. Gerd Wagner Frank Werner Tuncer Oren Floriano De Rango. 240c.

3. Neural Networks and Deep Learning. Michael Nielsen. 2019. 224c.

4. Наука про мережі: вступний курс. Девіс К. А., Фортунато С., Менцер Ф. 2022. 51 с.

5. Доставка майбутнього. Як самокеровані роботи, авто і дрони змінюють сучасну логістику. URL: https://forbes.ua/innovations/dostavka-maybutnogo-yak-samokerovani-roboti-avto-i-droni-zminyuyut-suchasnu-logistiku-14042023-13083 (дата звернення: 18.03.2024).

6. What Is ChatGPT Doing ... and Why Does It Work? Stephen Wolfram. 2023. 102c.

7. Deep Learning. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. 777c.

8. Human-level control through deep reinforcement learning. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves A., Riedmiller M., Fidjeland A., Ostrovski G., Petersen S., Beattie C., Sadik A., Antonoglou I., King H., Kumaran D., Wierstra D., Legg S., Hassabis, D. 2015. 518c.

9. Deep Learning with TensorFlow and Keras - Third Edition. Amita Kapoor, Antonio Gulli, Sujit Pal. 2022. 698c.

10. Python for Unity. URL: https://docs.unity3d.com/Packages/com.unity.scripting.python@2.0/manual/index.html (дата звернення: 01.05.2024).

11. Deep Reinforcement Learning in Unity: With Unity ML Toolkit. Abhilash Majumder. 2020. 572c.

12. ML-Agents Toolkit Overview. URL: https://unity-technologies.github.io/ml-agents/ML-Agents-Overview (дата звернення: 03.05.2024).

13. Reinforcement Learning: An Introduction Second edition, in progress Richard S. Sutton and Andrew G. Barto c 2014, 2015 54c.

14. Introduction to Unity ML-Agents: Understand the Interplay of Neural Networks and Simulation Space Using the Unity ML-Agents Package. Dylan Engelbrecht. 2023. 224c.

15. Competitive self-play with Unity ML-Agents. URL: https://www.gocoder.one/blog/competitive-self-play-unity-ml-agents (дата звернення: 06.05.2024).

16. Training intelligent adversaries using self-play with ML-Agents. URL: https://blog.unity.com/engine-platform/training-intelligent-adversaries-using-self-play-with-ml-agents (дата звернення: 07.05.2024).

17. Designing a Learning Environment. URL: https://github.com/Unity-Technologies/ml-agents/blob/develop/docs/Learning-Environment-Design.md agents (дата звернення: 08.05.2024).

18. MA-POCA. URL: https://github.com/Unity-Technologies/paper-ml-agents/tree/main/ma-poca (дата звернення: 12.05.2024).

19. Training Configuration File. URL: https://github.com/Unity-Technologies/ml-agents/blob/develop/docs/Training-Configuration-File.md#common-trainer-configurations (дата звернення: 13.05.2024).

20. Training Configuration File. URL: https://unity-technologies.github.io/ml-agents/Training-Configuration-File (дата звернення: 13.05.2024).

21. Learn Unity ML-Agents - Fundamentals of Unity Machine Learning. Micheal Lanham. 2018. 204c.

22. Testing Neural Networks and Machine Learning Models. URL: https://www.linkedin.com/pulse/testing-neural-networks-machine-learning-models-alexander-protasov (дата звернення: 14.05.2024).

23. Training your agents 7 times faster with ML-Agents. URL: https://blog.unity.com/engine-platform/training-your-agents-7-times-faster-with-ml-agents (дата звернення: 15.05.2024).

24. Hyperparameters tuning in ML-Agents. https://forum.unity.com/threads/hyperparameters-tuning-in-ml-agents.1142962 (дата звернення: 17.05.2024).

25. Best practices for ml-agents. URL: https://forum.unity.com/threads/asking-for-advice-and-best-practices-for-ml-agents-and-my-own-project.851569 (дата звернення: 18.05.2024).

26. Grebennik, I., Hubarenko, Y., Ananiev, M. (2022). Information Technologies for Assessing the Effectiveness of the Quarantine Measures. In: Sasaki, J., Murayama, Y., Velev, D., Zlateva, P. (eds) Information Technology in Disaster Risk Reduction. ITDRR 2021. IFIP Advances in Information and Communication Technology, vol 638. Springer, Cham. https://doi.org/10.1007/978-3-031-04170-9_11