КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Кваліфікаційна робота
на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

КОМП'ЮТЕРНА ГРА З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Виконала студентка 4-го курсу
Валерія КАЛІНІНА _____
(підпис)

Науковий керівник:
професор кафедри теоретичної кібернетики
доктор фіз.-мат. наук, професор
Анатолій ПАШКО _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теоретичної кібернетики

«____»_____2023 р., протокол № ____

Завідувач кафедри

Юрій КРАК _____
(підпис)

Київ - 2023

РЕФЕРАТ

Обсяг роботи 43 сторінки, 13 ілюстрацій, 17 джерело посилань.

ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, ГЛИБОКЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, ЕВРЕСТИЧНИЙ АЛГОРИТМ, ГЕНЕТИЧНИЙ АЛГОРИТМ, Q-НАВЧАННЯ, PYTHON, PYTORCH, OPENCV.

Об'єктом аналізу є три різні агенти штучного інтелекту для гри Тетрис: агент на основі евристичного алгоритму, агент на основі генетичного алгоритму та агент на основі Q-навчання.

Об'єктом розробки є гра Тетрис, яка створена для реалізації трьох агентів штучного інтелекту.

Метою дипломної роботи є дослідження та розробка трьох штучних ігрових агентів на основі евристичного алгоритму, генетичного алгоритму, Q-навчання та їх комплексна оцінка для гри Тетрис.

Методами розроблення є технології штучного інтелекту, використання та аналіз алгоритмів штучного інтелекту.

Інструментами реалізації є мова програмування Python та бібліотеки PyTorch та OpenCV.

Результом роботи є проведена комплексна оцінка параметрів кожного агента і досліджено потенційні причини впливу параметрів. За результатами експериментів, які порівнювали різні агенти, дійшли висновку, що агент на основі навчання з підсиленням має найкращу продуктивність в цілому. Крім того, агент на основі генетичного алгоритму і агент, що використовує глибоке навчання, також можуть досягти великої продуктивності з правильними параметрами.

ЗМІСТ

ВСТУП

Оцінка об’єкта розробки. Сьогоднішні дослідження показують, що штучний інтелект має значний потенціал для поліпшення повсякденного життя людей. Ця технологія знайшла своє застосування в різних сферах, таких як комп'ютерні науки, фінанси, медицина та діагностика.

Застосування штучного інтелекту стає все більш популярним серед компаній. Наприклад, автоматизовані магазини та доставка товарів дронами вже стали реалістичними завдяки цій технології. Голосові помічники також стали невід'ємною частиною нашого повсякденного життя. Крім того, штучний інтелект знайшов своє застосування у сфері відеоігор. Наприклад, варто згадати про успішний штучний ігровий агент під назвою AlphaGo, який переміг кількох професійних гравців.

Тетрис, як одна з найпопулярніших відеоігор, представляє виклик для розробки ефективних стратегій для гравців. Відсутність чітких правил та великий простір станів роблять його складним для кількісної оцінки та створення задовільних стратегій для людей. Проте, завдяки штучному інтелекту, можна розробити ігрового агента, який буде володіти ефективними стратегіями для гри в Тетрис.

Актуальність роботи. Актуальність даної роботи полягає в тому, що штучний інтелект і машинне навчання є активно розвиваючимися галузями і мають великий потенціал у багатьох сферах, включаючи відеоігри. Тетріс, як одна з найпопулярніших відеоігор, представляє складність для розробки ефективних штучних ігрових агентів через свої правила та простір станів.

Підстави для виконання роботи. Дослідження різних підходів до розробки штучних ігрових агентів для Тетрісу, має практичне значення. Це дозволяє зрозуміти, які методи можуть бути найбільш ефективними і як вони можуть бути використані для покращення ігрового досвіду.

Мета і завдання роботи. Метою роботи є дослідження та розробка трьох штучних ігрових агентів та їх комплексна оцінка для гри Тетрис:

1. Евристичний агент: Використання класичного методу евристики для отримання вищого рахунку в Тетрисі.
2. Генетичний агент: Застосування алгоритму, що моделює природний процес еволюції, для пошуку оптимального рішення в грі.
3. Q-навчання агент: Використання алгоритму навчання з підсиленням на основі глибокої мережі Q(DQN) для досягнення високої продуктивності в Тетрисі.

Об'єкт дослідження. Гра Тетрис.

Предмет дослідження. Розробка та оцінка трьох штучних ігрових агентів для гри Тетріс.

Засоби реалізації. Мова програмування Python, бібліотеки PyTorch, OpenCV, matplotlib, numpy.

Можливі сфери застосування: Агенти можуть бути використані для створення інтелектуальних комп'ютерних противників в іграх. Подальший розвиток та оптимізація штучних ігрових агентів може мати позитивний вплив на відеоігрову індустрію, а також на інші сфери, де застосовується штучний інтелект. Такі дослідження допомагають розширити наші знання про можливості і обмеження штучного інтелекту і сприяють розвитку нових інноваційних рішень та технологій.

1 ПРЕДМЕТНА ОБЛАСТЬ

1.1 Штучний інтелект

Штучний інтелект (ШІ) — це галузь інформатики, яка займається розробкою інтелектуальних машин, здатних виконувати завдання, які зазвичай потребують людського інтелекту. Системи штучного інтелекту створені для навчання на досвіді, розпізнавання закономірностей і прийняття рішень на основі вхідних даних. [1]

Основні алгоритми і методи штучного інтелекту включають: [2]

1. Машинне навчання: Метод, що дозволяє комп'ютерам навчатися на основі вхідних даних і вдосконалювати свою продуктивність без явного програмування. До основних підходів машинного навчання належать навчання з учителем, навчання без учителя та навчання з підсиленням.
2. Глибоке навчання: Підгалузь машинного навчання, яка використовує нейронні мережі з багатьма шарами для ефективної обробки великих обсягів даних. Глибокі нейронні мережі дозволяють отримувати високоякісні прогнози і розпізнавати складні зразки у даних.
3. Нейронні мережі: Модель, яка імітує роботу людського мозку і складається зі з'єднаних штучних нейронів. Нейронні мережі використовуються для класифікації, регресії, розпізнавання образів та багатьох інших завдань.
4. Генетичні алгоритми: Інспіровані природним процесом еволюції, генетичні алгоритми використовуються для розв'язання оптимізаційних проблем. Вони використовують поняття генетичного коду, схрещування, мутації та відбору для ефективного пошуку найкращого рішення в великому просторі можливих варіантів.

5. Логічне програмування: Підхід до програмування, де логічні правила і обмеження використовуються для вирішення проблеми. Пролог є однією з популярних мов програмування для логічного програмування.
6. Методи класифікації та кластеризації: Алгоритми, які використовуються для групування даних за схожістю. Класифікація визначає приналежність даних до певних категорій, тоді як кластеризація розділяє дані на групи без заздалегідь визначених категорій.
7. Обробка природної мови: Галузь, що займається розумінням і генерацією мови людей комп'ютерами. Вона використовується для автоматичного перекладу, аналізу настроїв, синтезу мови та інших задач, пов'язаних з обробкою тексту.
8. Підсилене навчання: Підхід до навчання агента, який взаємодіє з оточенням і вчиться на основі нагород і покарань. Цей метод широко використовується в робототехніці та в галузі відеоігор.

1.2 Застосування штучного інтелекту в іграх

Комп'ютерні ігри є однією з найпоширеніших і важливих областей застосування штучного інтелекту. Використання ШІ у комп'ютерних іграх дозволяє створювати ворожих агентів, які можуть змагатися з людьми або співпрацювати з ними. Також допомагає реалістичніше моделювати поведінку неігрових персонажів (NPC), опонентів або союзників, забезпечуючи їхню інтелектуальну поведінку.

Застосування штучного інтелекту в комп'ютерних іграх стає все більш популярним і дозволяє покращувати якість геймплею, створювати більш реалістичні та захоплюючі ігрові світи та забезпечувати нові виклики для гравців. Таким чином випробування власних здібностей шляхом змагання в іграх давно стало частиною людської культури.

Для досліджень у галузі штучного інтелекту ігри надають розгорнуту платформу для випробування механізмів навчання та порівняння результатів

штучного інтелекту з результатами гравців-людей. Більше того, ігри зазвичай мають фіксований набір правил, стандартизовані вхідні та вихідні дані. У поєднанні з можливістю обмеження складності гри, ігри є чудовим середовищем для досліджень у галузі штучного інтелекту.

Використання штучного інтелекту в іграх має різноманітні застосування. Наприклад, дослідники можуть створювати штучних противників для поліпшення геймплею та реалістичності ігрового досвіду. Крім того, ігри надають можливість вивчення механік гри та розроблення ефективних стратегій для досягнення поставленої мети.

Один з важливих аспектів використання штучного інтелекту в іграх полягає у порівнянні його результатів з результатами гравців-людей. Це дозволяє оцінити ефективність штучного інтелекту та визначити його переваги та обмеження порівняно з людським ігровим досвідом.

Крім того, ігри зазвичай дозволяють обмежити складність середовища, що є важливим для дослідження різних аспектів штучного інтелекту. Це дозволяє зосередитися на конкретних проблемах та розвивати нові алгоритми та методи. [3]

1.2.1 Класичні ігри

Дослідження з використання штучного інтелекту для вирішення класичних ігор мають багатолітню історію. Одним з перших успішних прикладів є вирішення гри "Шашки" Артура Самюела в 1959 році. Він використовував специфічну мережу, яка проводила аналіз дерева рішень на кожному ході і обирала дію з найбільшими шансами на успіх. Для покращення цього підходу Самюел впровадив таблиці підписів із групуванням пов'язаних параметрів, які навчалися на понад 100 000 записаних ходах. Цей підхід дав можливість програмі Chinook перемогти на Чемпіонаті світу з шашок в 1994 році. [4]

Іншою грою, в якій штучний інтелект перемагав людських чемпіонів, є настільна гра Backgammon. З використанням штучної нейронної мережі TD-Gammon відбувалося вивчення затримки в часі, що дозволяло програмі грати на рівні найкращих гравців-людей до 1992 року. Після кожного раунду проводився аналіз і визначався рух з найбільшим впливом на успіх гравця. Програма G. Тезауро навчалася за допомогою 1,5 мільйона раундів самої гри, використовуючи лише основні правила Backgammon. [4]

Однак, найскладнішим викликом для штучного інтелекту стала гра Go. Складна структура гри вимагала великої кількості модулів для різних аспектів гри. Це призводило до проблем, оскільки навіть один слабкий модуль міг негативно впливати на результат гри. Протягом тривалого часу вчені вважали, що досягнення сверхлюдського рівня в грі Go є неможливим. У 2002 році Мартін Мюллер дійшов висновку, що гра Go залишиться "колючкою в боці дослідників штучного інтелекту".

Проте згодом з'явилося багато програм комп'ютерного Го, які викликали гравців Го. Одним з найбільш вражаючих досягнень стала програма AlphaGo від DeepMind. У 2016 році AlphaGo змогла перемогти 18-разового чемпіона світу Лі Седоля з рахунком 4-1 в Сеулі. [6] Ці результати були досягнуті за допомогою поєднання двох великих глибоких нейронних мереж. Одна з мереж передбачала експертні ходи і навчалася за допомогою навчання з учителем та підсиленого навчання, а інша мережа оцінювала значення кожного стану гри. Під час фактичної гри обидві мережі використовувалися разом з алгоритмом пошуку в дереві Монте-Карло для вибору найкращого ходу. Ці результати продемонстрували потужність методів підсиленого навчання. Пізніше, у 2017 році, DeepMind випустила програму AlphaGoZero, яка навчалася виключно за допомогою підсиленого навчання, граючи проти себе. AlphaGoZero перевершила всі попередні програми комп'ютерного Го, здобувши перемогу з рахунком 100-0 проти AlphaGo і 89-11 проти AlphaGo Master. [5]

Ці успіхи в грі Go підкреслюють силу методів підсиленого навчання, оскільки AlphaGoZero навчилася не тільки тактиці гри без участі людей, але й розробила нові стратегії, розширивши знання про цю давню гру. [6]

1.2.2 Відео ігри

Застосування штучного інтелекту в ігровій індустрії стало однією з найважливіших та найцікавіших областей розвитку. Від самого початку існування відеоігор, розробники намагалися створити інтелектуальних противників, які можуть реалістично реагувати на дії гравця.

У ранніх відеоіграх, що почали з'являтися у 1970-х та 1980-х роках, противники зазвичай використовували прості алгоритми, такі як рух по певному шаблону або випадкові дії. Проте з розвитком технологій та появою потужних обчислювальних систем з'явилися нові можливості для застосування ШІ в іграх.

У 1990-х роках розробники почали використовувати більш складні алгоритми ШІ для створення інтелектуальних противників. Наприклад, у відомій серії файтингів Mortal Kombat гравці стикалися з противниками, які вміли виконувати різноманітні комбо-атаки та виявляти стратегічний ігровий геній.

У 2000-х роках ШІ відіграло важливу роль у створенні ігор з відкритим світом, де не головними персонажами були гравці, але також і непередбачувані противники та непередбачувані ситуації. Такі ігри, як Grand Theft Auto і The Elder Scrolls, використовують широкий спектр алгоритмів ШІ для моделювання реалістичного руху персонажів, їх соціальних взаємодій та прийняття рішень у великому світі гри.

У 2010-х роках розвиток ШІ в іграх продовжився з новими технологіями і алгоритмами. З'явилися ігри, які використовують машинне навчання та глибоке навчання для створення інтелектуальних систем. Наприклад, в 2011 році компанія Valve представила гру Dota 2 з ботами, що використовують навчання з підсиленням для розвитку своїх навичок гри та здатності виконувати складні стратегії.

Найвизначнішим досягненням використання ШІ в іграх стала перемога AlphaGo, розробленого компанією DeepMind, у відомій грі Го проти одного з найкращих гравців світу в 2016 році. Цей успіх продемонстрував потужність і можливості ШІ в складних стратегічних іграх.

В сучасних відеоіграх ШІ використовується для створення реалістичних та інтелектуальних противників, управління соціальною поведінкою великих груп персонажів, оптимізації штучного світу, підтримки гравцями інтерактивного геймплею та багато іншого.

Завдяки постійному розвитку технологій ШІ, можна очікувати, що майбутнє відеоігор буде ще більш захоплюючим і інтелектуально вдосконаленим. ШІ продовжує розширювати свої можливості і вносить значний вклад у покращення якості та реалізму відеоігрового досвіду для гравців.

У 2013 році, компанія DeepMind зробила великий крок у розвитку штучного інтелекту в ігровій сфері. Вони опублікували статтю про гру в ігри на приставці Atari2600, де застосували згорткову нейронну мережу та навчання з підкріпленням. Цей алгоритм зміг самостійно навчитися грати в кілька ігор з нуля, виявляючи надзвичайну продуктивність. [7]

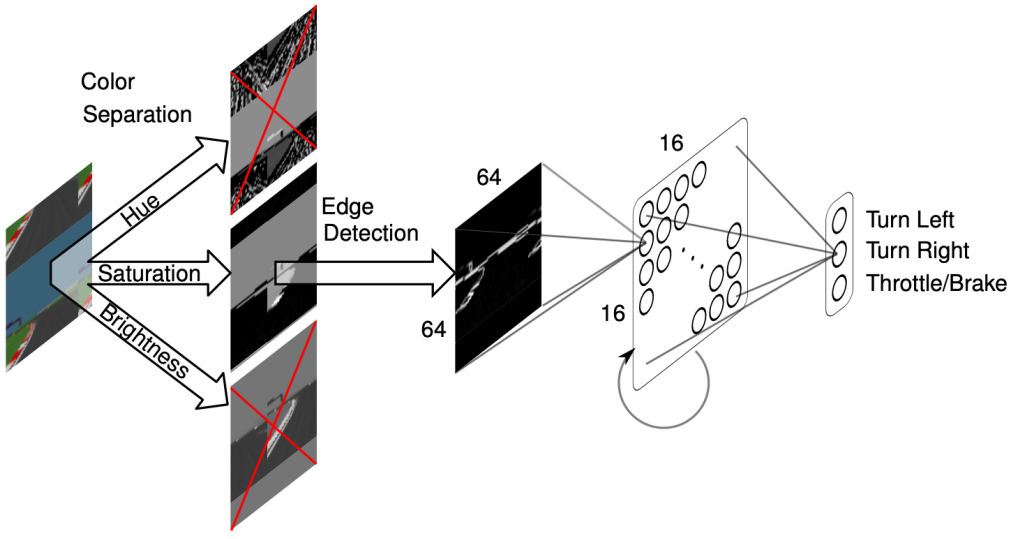

Рис.1 – Розташування контролера для візуального середовища TORCS

Також, було проведено дослідження використання штучного інтелекту в грі TORCS (The Open Racing Car Simulator). Швейцарська лабораторія штучного інтелекту IDSIA використала стиснуту рекурентну нейронну мережу (RNN) з більш ніж 1 мільйоном ваг, щоб навчити агента штучного інтелекту керувати автомобілем в грі. Відеозображення з погляду водія використовувалися як вхідні дані. Штучна нейронна мережа зберігала інформацію з попередніх зображень, щоб визначити швидкість зі статичного зображення.

Ці дослідження показали, що навчені контролери на основі RNN демонструють подібну продуктивність до контролерів, які мають повний доступ до функцій гри, таких як швидкість і положення на трасі. Було також виявлено, що зменшення розмірності вхідних даних RNN покращує процес навчання.

Дослідження продовжуються, і в майбутньому можна очікувати ще більшого розвитку штучного інтелекту в іграх. Щоб впоратися з викликом вирішення більш складних ігор стратегії в реальному часі за допомогою штучного інтелекту, дослідники розробили гру під назвою Deep Line Wars. Ця гра базується на грі Tower Line Wars з Warcraft III і вимагає використання різних стратегій для створення військ, захисту від ворожих одиниць і забезпечення рівноваги між ними для максимізації доходу. [8]

У цьому дослідженні автори адаптували алгоритми Q-навчання для використання в грі Deep Line Wars. Зображення гри не використовувалися як вхідні дані для згорткової нейронної мережі (CNN). Замість цього, створювалося абстрактне багатовимірне представлення поточного стану гри, яке перетворювалося в низькорозмірну карту теплових точок. Ця карта потім використовувалася для подальшого подання гри в мережу у вигляді піксельного зображення.

Хоча агенти, навчені з використанням алгоритмів Q-навчання, здатні грати в складні ігри краще, ніж агент, який використовує випадкову стратегію, вони поки що не змогли перевершити агента, який діє на основі правил і має схожу продуктивність з людським гравцем. Однак експерименти показали, що складні RTS-ігри можуть бути

успішно оволодіні алгоритмами Q-навчання. Виявлено, що складність управління та інтерпретації рухів миші є причиною меншої продуктивності агентів з Q-навчання порівняно з агентами на основі правил. [8]

Рис.2 – Deep Line Wars

Крім того, дослідження також зосередилися на іграх у першій особі. У дослідженні була використана адаптована версія гри Soldier of Fortune 2. Воно оцінило продуктивність різних алгоритмів машинного навчання в завданнях прискорення, адаптації напрямку руху і орієнтації, та стрибків.

Результати дослідження показали, що підхід з використанням штучної нейронної мережі (ANN), навчаного за допомогою експертної гри людини, перевершив інші алгоритми, такі як класифікатор наївного Байєса або алгоритм навчання дерева рішень. Однак продуктивність агентів ще не досягла прийнятного рівня, щоб використовувати їх як комп'ютерних ворогів у грі. Для подальшого покращення, дослідники використали методи "bagging" і "boosting". "Bagging" включає навчання ANN кілька разів на різних наборах даних для стабілізації класифікації. "Boosting" використовує взважування навчальних даних для

перенавчання неправильно класифікованих прикладів. Обидва методи покращили продуктивність агентів, причому "boosting" був ефективнішим у зниженні рівня помилок у всіх завданнях гри порівняно з "bagging".

Ці дослідження підтверджують можливість використання штучного інтелекту для вирішення складних завдань в ігровій сфері, таких як RTS і FPS ігри. Продовження роботи в цьому напрямку може призвести до подальшого покращення продуктивності агентів і створення більш складних та реалістичних ігрових досвідів для гравців.

У сфері розвитку штучного інтелекту в ігровій сфері, одним із викликів є навчання агентів грати в складні ігри, такі як ігри в жанрі FPS. Дослідники виявили, що навіть без людського досвіду агенти можуть навчитися грати в такі ігри за допомогою підсиленого навчання.

У одному з досліджень була створена спрощена гра у вигляді FPS, в якій агент, навчений алгоритмом табличного Sarsa(λ), зміг перемогти класичного супротивника, який працює на основі станових автоматів, у ситуації битви один на один. Замість використання сирого відеофрейму як вхідних даних для контролера, були створені сенсорні дані, такі як відстань до ворога, і передані агенту. Той самий навчальний середовище також використовувався для оцінки здатності агента навігації в лабіринтоподібній мапі. Виявилося, що штучний інтелект контролера зміг розробити стратегію навігації, але його продуктивність не досягала рівня стандартних алгоритмів пошуку шляху.

Згодом було проведено дослідження, що показало перевагу правило-заснованого навчання та ієрархічного підсиленого навчання над класичною версією навчання у вищезгаданому середовищі. Агент, навчений класичним підсиленням навчання, виявив здатність прицілитися на опонента і почати стріляти, але агенти, що навчалися ієрархічним і правило-заснованим підсиленим навчанням, демонстрували тактики, подібні до високопрофесійних людських гравців, зокрема, залучення до битви лише тоді, коли є висока ймовірність перемоги, і відступання, коли ворогів переважають чисельно.

Також існує інший підхід до моделювання адаптивних ШІ-агентів для ігор, запропонований К. Меррик. Вона використовує метод мотивованого підсиленого навчання для створення неігрових персонажів у MMORPG. Замість того, щоб передбачати конкретні завдання, які агент повинен навчитися, за допомогою функції винагороди, Меррик пропонує стимулювати пошук загальної компетентності. Цей підхід дозволяє гравцям адаптуватися до нових середовищ і розвиватися протягом гри, створюючи таким чином цікавіші та розважальніші досвіди гравця у віртуальному світі.

Ці дослідження підтверджують потенціал використання штучного інтелекту в ігровій сфері і вказують на можливість створення більш складних та реалістичних ігрових досвідів для гравців за допомогою алгоритмів підсиленого навчання.

2 ОГЛЯД ОБРАНИХ АЛГОРИТМІВ

Предметною областю даної роботи є застосування технік штучного інтелекту, зокрема евристичного алгоритму, генетичного алгортму та Q-навчання, в галузі розробки ігрових агентів для відеогри Тетрис, тому далі розглянемо їх.

2.1 Евристичний алгоритм

Евристичний алгоритм - це алгоритм, який використовує приблизні методи та правила, щоб знайти розв'язок проблеми, коли точний розв'язок не є можливим або дуже складним для знаходження в обмежений час. Він базується на евристичних методах, які є спрощеннями або емпіричними правилами, розробленими для полегшення процесу прийняття рішень. [9]

Основним принципом евристичних алгоритмів є використання спрощень та емпіричних правил, щоб зменшити обчислювальну складність задачі, але за ціну можливого втрати точності або оптимальності розв'язку. Це може бути прийнятно в ситуаціях, де швидкість вирішення проблеми має більший пріоритет, ніж точність або оптимальність розв'язку.

Евристичні алгоритми широко використовуються в різних областях, включаючи оптимізацію, штучний інтелект, машинне навчання та розпізнавання образів. Наприклад, в генетичних алгоритмах евристичні правила використовуються для знаходження приблизного оптимального розв'язку задачі шляхом емуляції еволюційних процесів. [10]

Однією з переваг евристичних алгоритмів є їхній здатність знаходити розв'язки в складних задачах, коли точні методи можуть бути неприйнятними з точки зору часу або ресурсів. Вони також можуть бути корисними в ситуаціях, коли немає доступу до повної інформації або коли проблема має багато параметрів, які складно оптимізувати точними методами.

Проте важливо зазначити, що евристичні алгоритми не гарантують знайдення оптимального розв'язку. Вони можуть давати приблизні або задовільні розв'язки, але не завжди найкращі. Також, результати евристичних алгоритмів можуть залежати від вибору початкових параметрів та евристик, які використовуються.

В цілому, евристичні алгоритми є потужним інструментом для вирішення складних проблем, коли точні методи не є практичними. Вони забезпечують швидкість вирішення за рахунок спрощень, але можуть не гарантувати оптимальність розв'язку.

2.1.1 Реалізація евристичного алгоритму

Оскільки ми розглядаємо евристичний алгоритм на основі агента для гри Тетрис, який базується на заданих правилах та евристичних оцінках для прийняття оптимальних дій маємо наступний розподіл алгоритму:

Ініціалізація:

- Створити початковий стан гри Тетрис.
- Визначити евристичні оцінки для різних аспектів гри, таких як висота стовпчиків, отвори, рядки, що можна заповнити, тощо.

Основний цикл:

- Отримати всі можливі наступні стани гри, які можуть виникнути в результаті різних дій (розташування тетраміно).
- Для кожного наступного стану виконати наступні кроки:
- Оцінити стан гри, використовуючи евристичні оцінки та правила гри.
- Зберегти оцінку для кожного наступного стану.
- Вибрати дію (розташування тетроміно), яке має найвищу оцінку відповідно до визначених евристик.
- Виконати вибрану дію та оновити стан гри.

Продовження гри:

- Повторювати кроки 2 досягнення кінцевого стану гри або до досягнення певної умови зупинки (наприклад, вичерпання обмеження часу чи кількості кроків).

Виведення результатів:

- Вивести фінальний стан гри, очки, розкладені рядки тощо.

Евристичний алгоритм для гри використовує задані правила та оцінки, які допомагають агенту приймати рішення щодо оптимальних дій. Наприклад, агент може оцінювати висоту стовпчиків гри, де більш високі стовпчики можуть мати негативний вплив на можливість заповнення рядків. Він також може оцінювати наявні отвори, де отвори між тетроміно можуть призводити до складнішого розташування наступних тетроміно.

Усі можливі наступні стани гри оцінюються, і найкраща дія вибирається на основі цих оцінок. Наприклад, агент може вибрати дію, яка мінімізує висоту стовпчиків або максимізує кількість заповнених рядків.

Цей евристичний алгоритм може бути постійно вдосконалюваний та доповнюваний новими евристиками, щоб досягти більш ефективного та точного прийняття рішень.

2.2 Генетичний алгоритм

Генетичний алгоритм зі зрізом (Genetic-Beam Search) є комбінацією двох популярних алгоритмів - генетичного алгоритму і пошуку зі зрізом (Beam Search). Цей алгоритм використовується для розв'язання оптимізаційних задач шляхом пошуку оптимального розв'язку у просторі можливих варіантів.

Генетичні алгоритми базуються на еволюційних принципах імітації природних процесів в живих організмах. Вони включають поняття популяції, хромосом, генів і функції пристосованості. Генетичний алгоритм працює за допомогою ітераційного

процесу, в якому кращі розв'язки виживають та розмножуються, створюючи нові покоління з кожною ітерацією. [11]

З іншого боку, пошук зі зрізом (Beam Search) є евристичним алгоритмом, який шукає розв'язок, обмежуючи кількість розглянутих можливих варіантів за допомогою певного параметру, називаного зрізом (beam). Він використовує локальну оптимізацію шляхом розгляду лише обмеженого набору потенційних розв'язків.

Генетичний алгоритм зі зрізом комбінує ці дві техніки. Він використовує генетичні оператори, такі як схрещування (комбінування хромосом) і мутація (зміна генів), для створення нових розв'язків у кожному поколінні. За допомогою функції пристосованості відбираються кращі розв'язки для виживання.

Однак, замість того, щоб створювати повну популяцію, генетичний алгоритм зі зрізом обмежує кількість розглянутих розв'язків за допомогою параметру зрізу. Це дозволяє зберегти лише обмежену кількість найкращих розв'язків на кожній ітерації, зменшуючи просторову та часову складність алгоритму.

Генетичний алгоритм зі зрізом поєднує сильні сторони обох методів - генетичного алгоритму та пошуку зі зрізом. Він може бути особливо корисним у випадках, коли простір розв'язків великий або коли точні методи оптимізації є дорогими з точки зору обчислювальних ресурсів.

2.2.1 Реалізація генетичного алгоритму зі зрізом

Оскільки ми розглядаємо генетичний алгоритм зі зрізом на основі агента для гри Тетрис, який поєднує елементи еволюції і генетики для пошуку оптимального рішення, то розподіл алгоритму може бути наступним:

Ініціалізація популяції:

- Визначити параметри генетичного алгоритму, такі як розмір популяції, кількість поколінь, ймовірності мутації і схрещування тощо.

- Згенерувати початкову популяцію індивідуальних рішень (стратегій гри), наприклад, випадковим чином.

Основний цикл генетичного алгоритму:

- Для кожного покоління виконати наступні кроки:
- Оцінити кожного індивіда в популяції, використовуючи оцінкову функцію, яка вимірює якість стратегії гри.
- Відібрати найкращих індивідів, використовуючи певний метод відбору (наприклад, турнірний відбір).
- Застосувати оператори схрещування і мутації до відібраних індивідів для генерації нових нащадків.
- Замінити стару популяцію новою популяцією, включаючи як старих індивідів, так і новостворених нащадків.

Продовження циклу генетичного алгоритму:

- Повторювати крок 2 досягнення заданої кількості поколінь або до досягнення певної умови зупинки (наприклад, достатньої якості стратегій гри).

Виведення результатів:

- Вивести найкращу стратегію гри, яка була знайдена після закінчення генетичного алгоритму.
- Вивести іншу статистику, таку як середній рейтинг популяції, зміна рейтингу з покоління в покоління, тощо.

Головна ідея генетичного алгоритму полягає в емуляції природного відбору, де кращі рішення мають більшу ймовірність пережити та передати свої хороші властивості наступному поколінню. Застосовуючи оператори схрещування і мутації, алгоритм розширює простір пошуку та можливість знаходження оптимального рішення.

У контексті гри Тетрис, генетичний алгоритм зі зрізом може генерувати та оцінювати різні стратегії гри, які враховують фактори, такі як висота стовпчиків,

кількість заповнених рядків, наявність отворів тощо. Шляхом відбору та еволюції найкращих стратегій гри, генетичний алгоритм здатний покращувати результати гри з кожним поколінням, наближаючи до оптимальної стратегії.

Використання генетичного алгоритму зі зрізом у грі відображає потужний підхід до вирішення проблеми прийняття рішень в складних вузьких галузях, де традиційні методи можуть бути недостатніми. Цей алгоритм дозволяє знайти оптимальні стратегії гри, що сприяє досягненню кращих результатів у грі Тетрис.

2.3 Глибоке Q-навчання

Глибоке Q-навчання (Deep Q-Learning) є алгоритмом з підсиленим навчанням, який використовує глибокі нейронні мережі для навчання агента приймати рішення в середовищі з великою кількістю можливих станів і дій. Він є розширенням класичного Q-навчання, яке використовує таблицю Q-значень для збереження оцінок дій в кожному стані. [13]

Основна ідея глибокого Q-навчання полягає в тому, що використовуються глибокі нейронні мережі (зазвичай засновані на згорткових мережах) для наближення функції Q-оцінки. Функція Q-оцінки визначає, яка дія має найбільшу винагороду в даний момент і є основою для прийняття рішень агентом. Глибокі нейронні мережі навчаються апроксимувати функцію Q-оцінки шляхом навчання на великому наборі даних, отриманих в процесі взаємодії агента з середовищем.

Процес навчання глибокого Q-навчання включає в себе ітерації, в яких агент взаємодіє з середовищем, спостерігаючи поточний стан і вибираючи дію за допомогою стратегії вибору дії (наприклад, ε-жадний підхід). Після кожної дії агент отримує винагороду і спостерігає новий стан. Оцінки Q-функції оновлюються на основі отриманих даних, використовуючи метод градієнтного спуску.

Однак, глибоке Q-навчання має свої особливості і виклики. Один з головних викликів полягає в управлінні компромісом між зберіганням досвіду та розвиванням

нових знань. Це може бути досягнуто за допомогою підхідів, таких як досвідна повторна гра (experience replay), яка зберігає і використовує попередні дані для навчання мережі.

Крім того, глибоке Q-навчання також може стикатися з проблемою "прокляття розмірності" (curse of dimensionality), особливо в великих середовищах, коли кількість можливих станів і дій дуже велика. Для подолання цієї проблеми використовуються різні техніки, такі як функції нагороди, які дозволяють зменшити простір станів або методи зменшення розмірності вхідних даних.

Глибоке Q-навчання здобуло значну популярність і застосовується в різних сферах, включаючи робототехніку, ігрову індустрію, фінанси та багато інших. Воно може вирішувати складні завдання з прийняття рішень в середовищах зі значною степенем невизначеності та навчатися вирішувати проблеми шляхом взаємодії з навколишнім середовищем. [14]

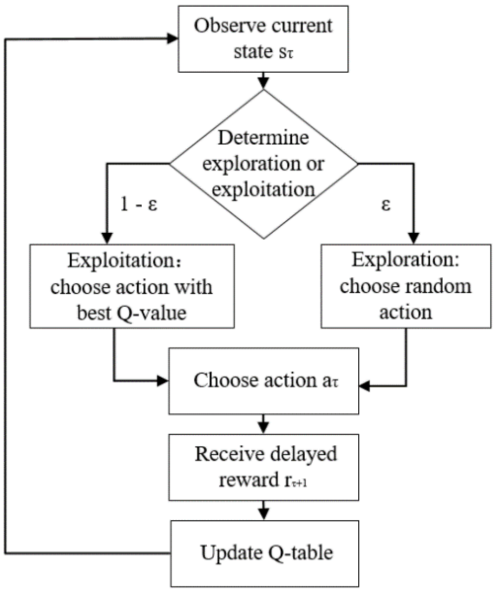

Рис.4 – алгоритм Q-навчання

2.3.1 Реалізація алгоритму Q-навчання

Оскільки ми розглядаємо Глибоке Q-навчання на основі агента для гри Тетрис, який є потужним алгоритмом навчання з підсиленням маємо наступний розподіл алгоритму:

Ініціалізація мережі Deep Q-Network (DQN):

- Визначити архітектуру глибокої нейронної мережі, яка буде використовуватись для наближення функції Q-оцінки.
- Задати початкові ваги мережі.
- Збирання даних та навчання:
- Взаємодія з середовищем Tetris, де агент здійснює дії і отримує винагороду за кожен крок.
- Збереження даних про стан середовища, зроблені дії, отриману винагороду та наступний стан.
- Оновлення функції Q-оцінки за допомогою зворотного поширення помилки на основі отриманої винагороди.
- Використання методу Q-навчання для оновлення Q-оцінки, що дозволяє агенту навчитись вибирати оптимальні дії відповідно до очікуваної винагороди.

Політика розгляду (Exploration Policy):

- Для забезпечення розвідки простору станів і дій, агент може використовувати певну політику розгляду, наприклад, епсилон-жадібну стратегію. Це означає, що агент випадковим чином вибирає неоптимальні дії з ймовірністю епсилон, щоб досліджувати нові можливості.

Оновлення мережі:

- Періодично оновлювати ваги мережі на основі накопичених даних.
- Використовувати алгоритм оптимізації, такий як стохастичний градієнтний спуск, для оновлення ваг мережі на основі втрати між прогнозованими Q-

значеннями та насправді отриманими винагородами.

Повторення:

- Продовжувати взаємодіяти з середовищем, збирати дані і навчати мережу в ітераційному режимі.
- Повторювати процес збору даних та навчання, щоб поступово покращувати стратегію агента і досягти більш високих винагород.

Глибоке Q-навчання є потужним алгоритмом, оскільки використовує глибоку нейронну мережу для наближення функції Q-оцінки. Це дозволяє агентові оцінювати очікувану винагороду для різних станів і дій у грі. Завдяки зворотному поширенню помилки та оновленню ваг мережі, агент може навчитись ефективно вибирати дії, що максимізують його винагороду. Використання глибокого Q-навчання у грі Тетрис дозволяє створити агента, який здатен досягати високих результатів у грі та вдосконалюватись з кожним навчальним циклом.

2.4 Принципові відмінності методів

Евристичний алгоритм, генетичний алгоритм і Q-навчання - це різні методи і підходи до розв'язання задач у галузі штучного інтелекту. Основні принципи та принципові відмінності між цими методами можуть бути пояснені наступним чиної : Евристичний алгоритм:

- Принцип: Евристичний алгоритм використовує правила, експертні знання та емпіричний досвід для розв'язання задачі. Він базується на виборі найбільш перспективних шляхів або рішень на основі доступної інформації.
- Відмінності: Евристичний алгоритм не гарантує знаходження оптимального рішення і зазвичай працює на основі експертного знання, яке може бути обмеженим або піддається певним перекосам. Він підходить для задач, де можливість перебору всіх варіантів є недоцільною з точки зору обчислювальної складності.

Генетичний алгоритм:

- Принцип: Генетичний алгоритм моделює процес природного відбору та генетики для пошуку оптимальних рішень у великому просторі можливих варіантів. Він працює шляхом створення популяції рішень, їх схрещування та мутації, оцінки пристосованості та відбору найкращих рішень для наступного покоління.
- Відмінності: Генетичний алгоритм зазвичай використовується для оптимізаційних задач, де потрібно знайти найкращі параметри або комбінації. Він може працювати у великому просторі можливих рішень, але його ефективність залежить від вибору операторів схрещування та мутації, а також від якості функції пристосованості.

Q-навчання:

- Принцип: Q-навчання - це метод навчання з підкріпленням, який дозволяє агенту навчитися приймати рішення в стохастичному середовищі. Агент вчиться оцінювати якість дій у різних станах середовища (q-значення) на основі отриманої винагороди та навчального коефіцієнта.
- Відмінності: Q-навчання засноване на ідеї навчання на основі винагороди та оновлення q-значень. Воно вимагає взаємодії агента з середовищем та збереження таблиці q-значень. Q-навчання може використовуватися для навчання агентів, які приймають рішення в динамічних ігрових середовищах.

3 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

3.1 Мова Python та бібліотеки PyTorch, OpenCV

Python – мова програмування, яка інтерпретується на високому рівні. Головна мета створення цієї мови полягала в тому, щоб програмний код був близьким до математичних виразів. Python ідеально підходить для початківців, оскільки його код легко читається та зрозуміти. Водночас, він дозволяє використовувати як елементи функціонального програмування, так і елементи об'єктно-орієнтованого програмування.

PyTorch - відкрита бібліотека машинного навчання, яка базується на бібліотеці Torch. Вона застосовується для розв'язання завдань комп'ютерного бачення та обробки природної мови. Розробку PyTorch в основному веде група дослідників зі штучного інтелекту в компанії Facebook. Ця бібліотека є вільним програмним забезпеченням, яке випускається під ліцензією Modified BSD. Хоча основний інтерфейс PyTorch реалізований на Python, вона також надає зовнішній інтерфейс для використання з C++.

OpenCV - бібліотека з відкритим кодом, яка надає набір функцій та алгоритмів для комп'ютерного зору, обробки зображень та чисельних обчислень. Ця бібліотека дозволяє виконувати обробку і аналіз зображень, зокрема розпізнавання об'єктів на фотографіях (наприклад, облич, фігур, тексту), відстежування руху об'єктів, перетворення зображень, використання методів машинного навчання та виявлення загальних елементів на різних зображеннях.

3.2 Інші інструменти реалізації

У роботі також були використані такі інструменти:

Matplotlib – бібліотеки для візуалізації графіків.

Numpy – математичний пакет для роботи з масивами. Також містить багато математичних операцій.

argparse - це модуль, який дозволяє обробляти аргументи командного рядка.

4 РЕАЛІЗАЦІЯ

4.1 Огляд на гру Тетрис

Тетрис - це класична відеогра, яка була випущена в середині 80-х років. Гра передбачає розташування семи різних типів блоків, також відомих як "Тетраміни" (Рис.5)

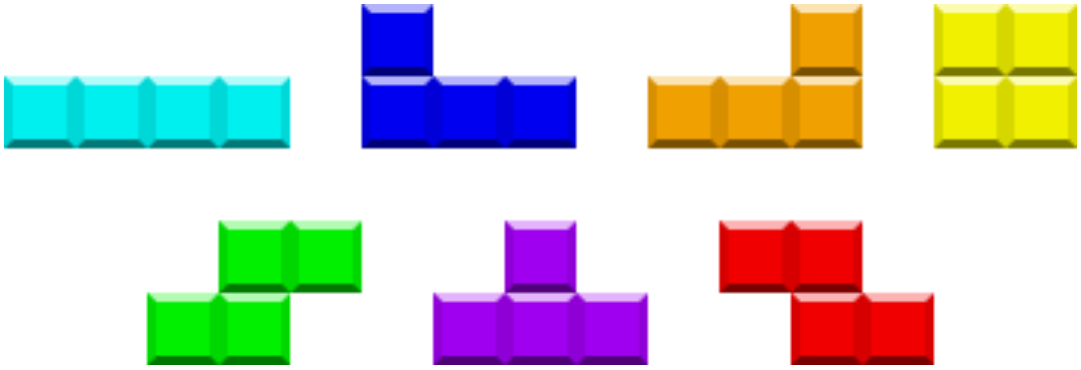

Рис.5 - Тетраміни

для створення ліній на ігровому полі. Блоки падають зверху вниз на ігрове поле і можуть лише переміщатися вліво або вправо гравцем. Кожного разу, коли блок розміщується, новий випадковий блок починає опускатися з верхньої частини екрана. Залежно від варіанту гри, ігрове поле може або не може знати, який буде наступний тип блоку. Блоки падають на сітку розміром 20 квадратів висотою і 10 квадратів шириною. Кожного разу, коли лінія повністю заповнюється, вона зникає, розкриваючи нижню лінію. Коли стопка блоків досягає верху сітки, гра закінчується.

Гра Тетріс є абсолютно непереможною, оскільки вона сильно залежить від послідовності блоків, які генеруються, тому будь-яке незграбне поєднання блоків, таких як S і Z, призведе до швидкого та необхідного завершення гри. [16]

4.2 Правила і логіка

Метою гри Тетрис є набрати якомога більше очок, шляхом ефективного розміщення фігур і заповнення рядків без прогалин. Гра розвиває швидкість реакції, просторове мислення та вміння приймати рішення на ходу.

Агент Тетриса став класичним застосуванням технології навчання з підсиленням в галузі ігрових агентів. Як довгострокова гра, у Тетриса є чітке визначення простору станів та відносно простий механізм гри.

Дії. Дія у Тетрісі складається з двох параметрів: позиції падіння та кількості обертів. Позиція падіння визначає кінцеве положення поточно падаючого тетроміно, а кількість обертів визначає кінцеву орієнтацію тетроміно.

Стани. Подібно до більшості відеоігор, важко точно кількісно оцінити і моделювати стан Тетріса.

Найпростіший і зрозумілий спосіб представлення стану гри в Тетрісі - використання сітки розміром N × M. Порожнє положення (тобто без блоків) сітки позначається як 0, а зайняте положення позначається іншими числами (наприклад, 1, 255).

Якщо висота стовпця перевищує N після падіння блоку, гра закінчується і оцінюються кінцеві бали гри. Кінцевий рахунок гри розподіляється на три частини: бали за тетраміно, бали за видалені лінії та загальний бал.

Опишемо визначення, які використовуються в наступних алгоритмах:

A - простір дій

a - вибрана дія

S - поточний стан

S’ - простір можливих наступних станів

S’(S, a) - наступний стан після виконання дії a у стані S

Q(S, a) - значення Q для виконання дії a в стані S

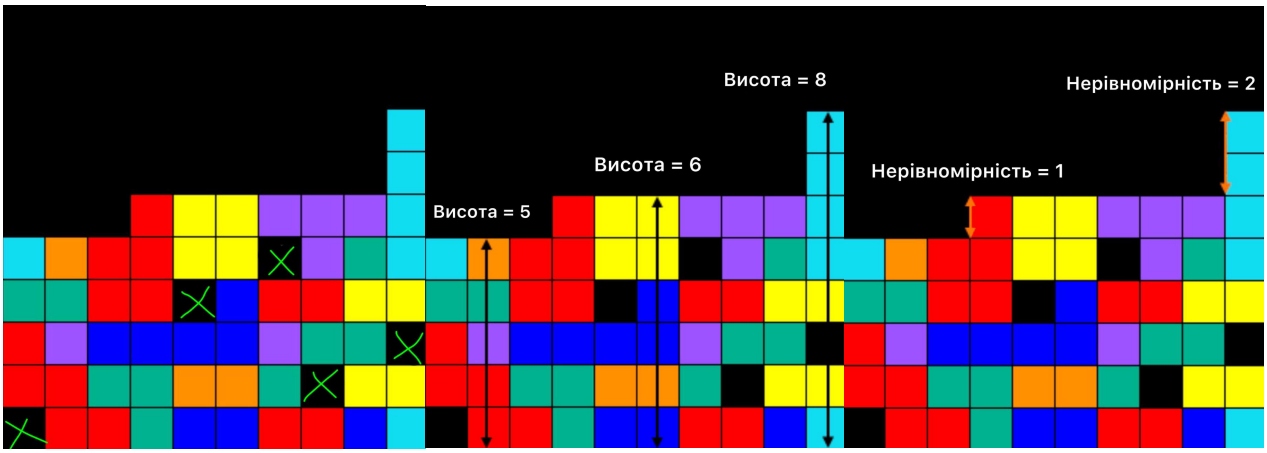

Рис.6.1. Рис.6.2 Рис.6.3

Дірки. Ми визначаємо пусту позицію як дірку, якщо вона оточена ненульовими значеннями або межею дошки, Рис.6.1.

Висота блоку. Відстань від найнижчого ненульового числа до найвищого ненульового числа кожного стовпця є висотою стовпця, Рис.6.2.

Нерівномірність. Список різниць між висотами стовпців двох сусідніх стовпців, Рис.6.3.

4.3 Агент на основі евристичного алгоритму

Евристичний алгоритм є класичним методом вирішення пошукових задач, який схожий на нашу проблему Тетрису. Тому він надає базову, але ефективну ідею для отримання вищого рахунку в грі, включаючи два кроки. Перший крок - розраховує значення h для кожної дії з падаючим тетраміно, за допомогою нашої евристичної функції. Другий крок - вибирає найкращу дію згідно зі значенням h, щоб ми могли максимізувати рахунок нашого агента. Евристична функція має наступний вигляд:

$$h(f_{1:8}) = \sum_{i=1}^{i=8} w_i * f_i$$

Для створення кращої евристичної функції вибрано 8 параметрів на основі двох частин: властивостей ігрової дошки та S'(S, a).

4.4 Агент на основі генетичного алгоритму

Генетичний пошук - це алгоритм, який шукає оптимальне рішення, симулюючи природний процес еволюції. Цей агент, перетворює процес пошуку рішення на процес, подібний до кросовера та мутації хромосом у біологічній еволюції. Основний процес агента виглядає так:

Крок 1: Випадковим чином генеруємо 7 хромосом (за рівномірним розподілом) і позначаємо їх як перше покоління.

Крок 2: Використовуючи поточне покоління як евристичну функцію системи, запускаємо гру. Для кожного падаючого елемента використовуємо покоління, щоб отримати найкращу дію. Після завершення гри записуємо рахунок.

Крок 3: Застосовуємо кросовер хромосом та генетичний алгоритм звуження до покоління, щоб сформувати нове покоління. Під час кросовера хромосом, кожний ген хромосоми має шанс α отримати ген від одного з батьків і шанс 1-α отримати середнє значення відповідних генів двох батьків. Під час генетичного звуження нове покоління містить β кращих хромосом з попереднього покоління і 1-β одноточковий кросовер між кращими хромосомами попереднього покоління. Тимчасово, на цьому етапі є шанс γ мутації гена.

Крок 4: Повторюємо Кроки 2 і 3, щоб отримати кілька раундів еволюції для покоління. Порівнюючи отримані рахунки для кожного покоління, вибираємо найкраще покоління і використовуємо хромосоми як параметри остаточної евристичної функції для запуску Тетрису.

4.5 Агент на основі Q-навчання

Подібно до більшості відеоігор, точна кількісна характеристика та моделювання стану Тетрису є складними завданнями, тому уважно вибраємо чотири ознаки для представлення стану гри S'(S, a):

• очищені лінії: кількість ліній, які будуть очищені після виконання дії a у станіS

• отвори: кількість отворів

• нерівність: сума нерівностей

• висота блоків: сума висоти блоків у кожному стовпчику.

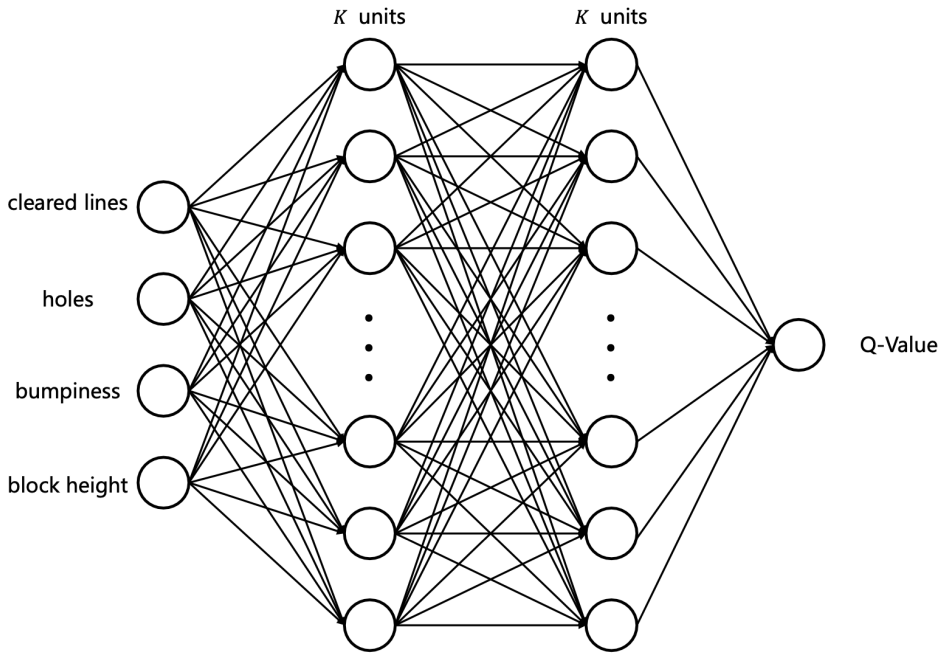

Рис. 7 – MLP для Q-навчання

Ми створюємо просту MLP (див. рис. 7), щоб передбачити значення Q-функції на основі чотирьох вхідних ознак. MLP має два прихованих шари, кожний з них має K вузлів. Ми досліджуємо вплив K на продуктивність.

4.6 Налаштування

Для оцінки трьох методів агента Тетрису встановлено N = 10, M = 20 (тобто сітка розміром 10×20) як тестове середовище.

Метрика оцінки. Щоб оцінити ефективність методів реалізації агента Тетрису, вибрано три критерії оцінки для кількісної оцінки продуктивності різних методів.

Детальне визначення трьох метрик:

• Тетраміно. Тетраміно - це кількість Тетраміно, які впали протягом всієї гри.

• Рядки. Рядки - це кількість видалених рядків протягом всієї гри.

• Очки. Для розрахунку кінцевих очок, для кожного падаючого блоку ми обчислюємо поточний результат за допомогою такого методу:

$$currentScore = 1 + C^2 * N$$

де C представляє кількість видаляються рядків на поточному кроці, а N - ширина сітки. Після отримання результату для кожного кроку, ми обчислюємо Очки за таким рівнянням:

$$Scores = \sum_{i=1}^{gameOver} currentScore_i$$

4.7 Результати для агенту на основі евристичного алгоритму

Для максимізації продуктивності агента, експериментуємо з різними наборами ваг для всіх восьми ознак. Для кожного набору ваг ми запускаємо 300 ігор і обчислюємо середні значення їх результатів - очок, кількості тетраміно та кількості рядків. Як показано на Рис. 8, не спостерігається значної різниці у продуктивності агента за методом евристики з різними вагами для ознаки 1. Інші ознаки, пов'язані зі

S', схожі між собою. Незалежно від вибору ваг для ознак 1-4, рахунок в грі коливається навколо 950 (середнє значення - 937,6458). Однак, порівняно з ознаками, пов'язаними зі S', ознаки, пов'язані зі S'(S, a), мають більший вплив на продуктивність агента за методом евристики. На відміну від ознак, пов'язаних із дошкою (тобто ознак 1-4), існує значна різниця між різними вагами ознак, пов'язаних із діями (тобто ознак 5-8). Ознака 5, пов'язана з дією "очищення рядків", має середнє значення рахунку 945,475 при позитивній вазі, але незадовільно при негативній вазі. Беручи до уваги властивість того, що f5 завжди більше 0, можлива причина полягає в тому, що позитивна вага робить f5 значною частиною в обчисленні евристичної функції. Це змушує агента віддавати перевагу видаленню рядків, якомога більше. Якщо вага має значне від'ємне значення, агент не буде прагнути видаляти рядки. Аналогічно, агент добре впорається, коли він намагається видалити більше отворів на дошці та утримувати блоки плоскими (ознаки "кількість отворів" та "нерівномірність").

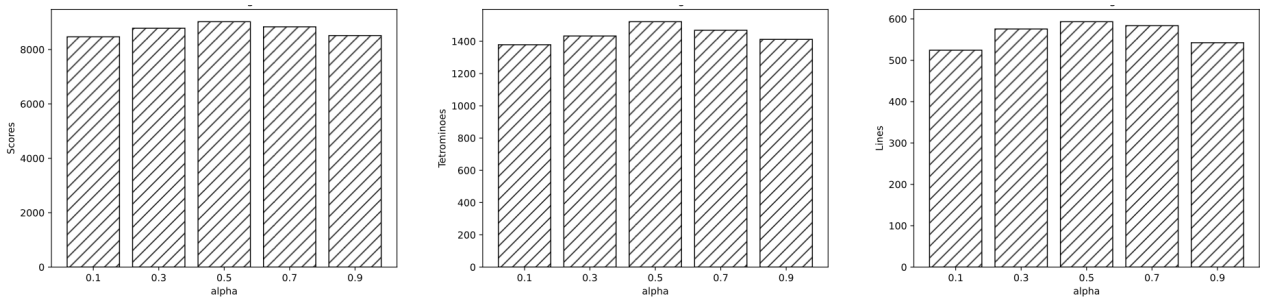

Рис.8 - Евристичний вигляд для максимальної висоти дошки з різними значеннями w1 на рахунки, тетраміно та лінії.

4.9 Результати для агенту на основі генетичного алгоритму

Ми оцінюємо продуктивність агента на основі генетичного алгоритму з різними значеннями α, β і γ. Для кожного випадку ми встановлюємо обмеження часу 5 хвилин на одну гру. Запускаємо 300 ігор для кожного випадку і обчислюємо середні значення рахунку в кінці. З наших експериментів ми встановили, що при фіксованих значеннях β та γ найкращі результати за рахунком, кількістю рядків і тетрамінами досягаються при α = 0,5, і зменшуються при збільшенні або зменшенні α на Рис.9

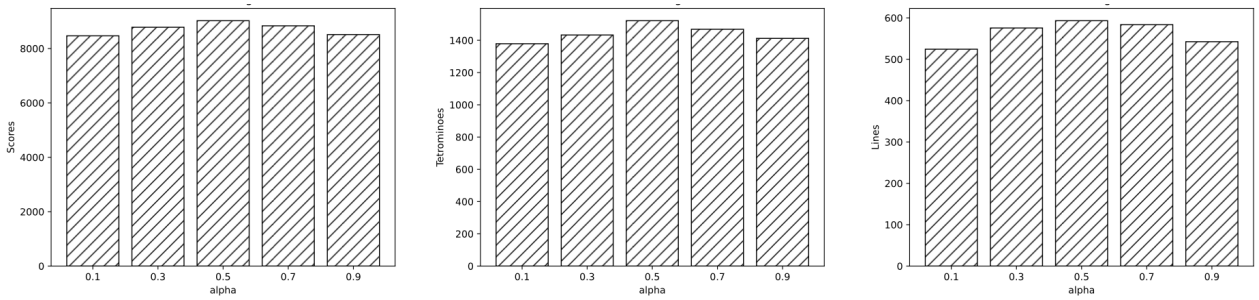

Рис.9 - Рахунки, тетраміно та лінії, коли β та γ фіксовані.

Потенційна причина такого результату полягає в тому, що α визначає швидкість відбору кращого покоління та схрещування хромосом. У цьому випадку краще мати 50% ймовірність схрещування хромосом. При фіксованих значеннях α та γ найкращі результати за рахунком, кількістю рядків і тетрамінами досягаються при β = 0,2, і зменшуються при збільшенні β на Рис. 10

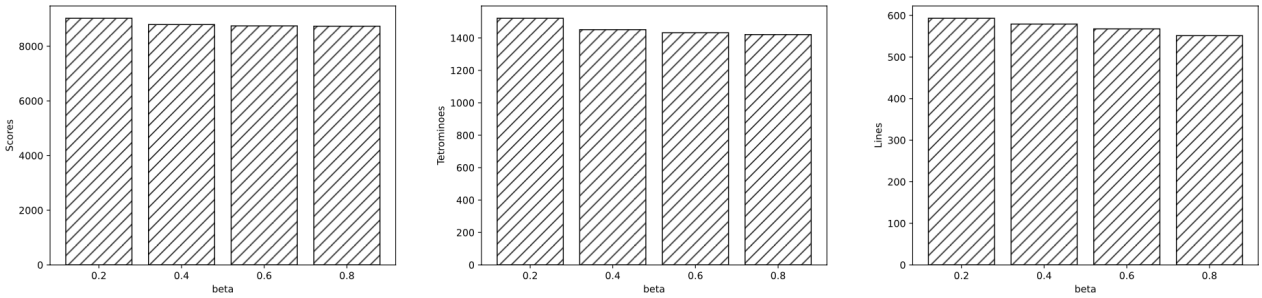

Рис.10 - Рахунки, тетраміно та лінії, коли α та γ фіксовані.

Потенційна причина цього результату полягає в тому, що β визначає ймовірність того, що хромосома буде прямою спадкоємцем одного з батьків, тому чим менше значення β, тим більша ймовірність того, що хромосома буде середнім значенням генів двох батьків. При фіксованих значеннях α та β найкращі результати за рахунком, кількістю рядків і тетромінами досягаються при γ = 0,1, і різко зменшуються при збільшенні γ (показано на Рис. 11).

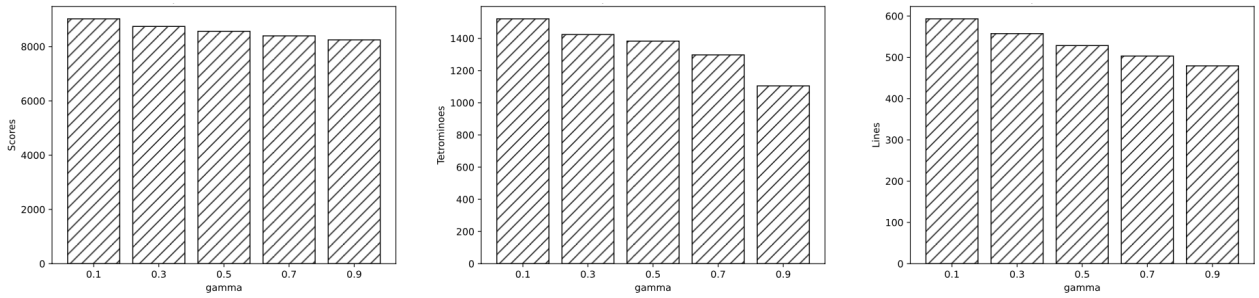

Рис.11 - Рахунки, тетраміно та лінії, коли α та β фіксовані.

Потенційна причина цього може бути в тому, що γ відображає швидкість мутації. Помірна мутація може призвести до отримання кращої хромосоми, проте при високій ймовірності випадкових змін результат може бути негативним.

4.10 Результати для агенту на основі Q-навчання

Як показано на Рис.12, ми оцінюємо продуктивність MLP з різною кількістю одиниць K. Для кожної моделі ми запускаємо 3000 ігор і обчислюємо середні значення рахунку. Однак модель MLP32 досягає такої великої продуктивності, що майже не доходить до умов кінця гри, описаних у розділі II, тому ми визначаємо його як inf. Усі MLP, які складніші за MLP32, мають гіршу продуктивність, і продуктивність зменшується зі збільшенням складності моделі. Потенційна причина полягає в тому, що вхідний вектор ознак має всього чотири виміри, і складні моделі зазнають

перенавчання, тому продуктивність погіршується. MLP16 працює погано, оскільки має недостатньо параметрів для відповідності станам та діям в грі.

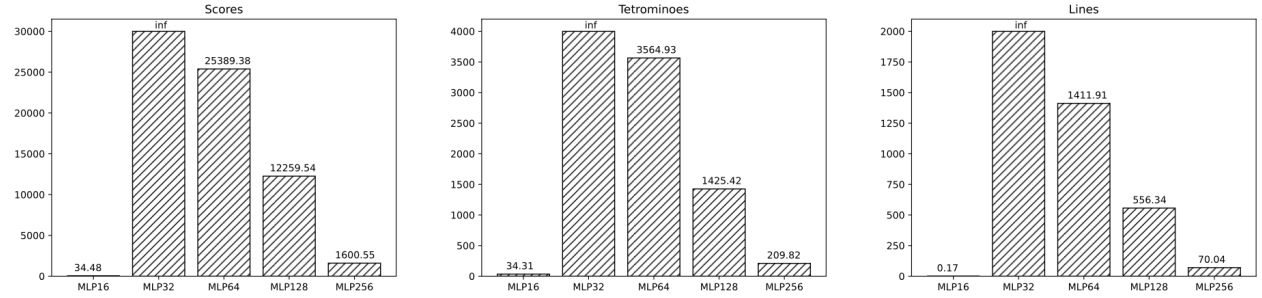

Рис.12 - Ефект кількості нейронів K в багатошаровому перцептроні (MLP) на рахунок, тетраміни та лінії

4.11 Порівняння

Оцінюємо продуктивність трьох агентів, фіксуючи кількість падаючих тетраміносів. Для кожного агента ми запускаємо 500 ігор з 500, 1000, 1500, 2000, 2500 тетраміносами відповідно, щоб отримати середній рахунок. Як показано на Рис.13,

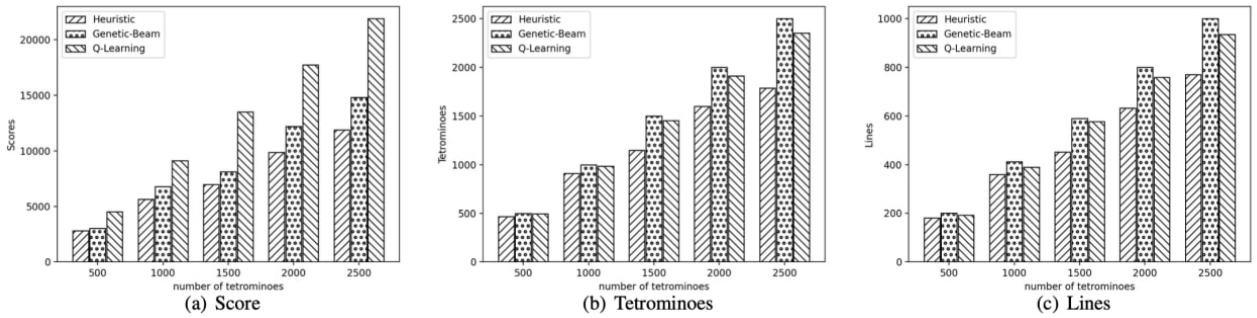

Рис.13 – Продуктивність 3 агентів за різних умов.

агент на основі Q-навчання в цілому отримує вищі рахунки, ніж інші методи. Зі зростанням кількості тетраміносів розрив у рахунках між агентом на основі Q-навчання та іншими агентами стає помітнішим. Однак, коли кількість тетрамін досягає 1500 або більше, є помітна різниця у середній кількості завершених

тетраміносів та вилучених ліній між агентом на основі евристики та іншими агентами. Це свідчить про те, що агент на основі евристики не є достатньо ефективним для виконання вимог. Агент на основі генетичного алгоритму зазвичай виживає найдовше і вилучає більше ліній, хоча його рахунок загалом нижчий, ніж у Q-навчання. Згідно з визначенням рахунків, можлива причина полягає в тому, що агент на основі генетичного алгоритму намагається видаляти лінії на кожному кроці, тоді як агент на основі Q-навчання намагається накопичувати тетраміноси, а потім видаляти кілька ліній за один крок, щоб отримати вищий рахунок.

4.12 Висновок до результатів

Ми провели комплексну оцінку параметрів кожного агента і дослідили потенційні причини впливу параметрів. За результатами експериментів, які порівнювали різні агенти, можна зробити висновок, що агент на основі Q-навчання має найкращу продуктивність в цілому, з досягненням середнього рахунку 21887,584 і середньої кількості видалених рядків 934,67 при фіксованій кількості тетрамінів у 2500. Крім того, агент на основі генетичного алгоритму і агент з евристичним підходом, також можуть досягти великої продуктивності з правильними параметрами.

5 ВИСНОВКИ

Метою роботи було дослідження та розробка трьох штучних ігрових агентів на основі евристичного алгоритму, генетичного алгоритму, Q-навчання та їх комплексна оцінка для гри Тетрис

Для виконання поставленої мети було виконано:

1. Підібрано математичні інструменти для вирішення поставленої задачі;
2. Підібрано інструменти для реалізації та навчання;
3. Визначено та реалізовано необхідні кроки для реалізації штучних агенті для гри;
4. Виконано навчання;
5. Проведене порівняння та аналіз результатів;
6. Було розроблено три штучних інтелектуальних агента для гри Тетрис, включаючи евристичний алгоритм, генетичний алгоритм і навчання з підсиленням з використанням глибоких нейронних мереж. Ми провели комплексну оцінку параметрів кожного агента і дослідили потенційні причини впливу параметрів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Штучний інтелект [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82.
2. ЩО ТАКЕ ШТУЧНИЙ ІНТЕЛЕКТ: ІСТОРІЯ, ВИДИ ТА СКЛАДОВІ [Електронний ресурс] – Режим доступу до ресурсу: https://gigacloud.ua/blog/navchannja/scho-take-shtuchnij-intelekt-istorija-vidi-ta-skladovi.
3. AI in Gaming [Електронний ресурс] – Режим доступу до ресурсу: https://www.arm.com/glossary/ai-in-gaming.
4. Lucci S., Kopec, D.: Artificial Intelligence in the 21st Century. 2013, ст. 482–525.
5. Samuel A. : Some studies in machine learning using the game of checkers. In: IBM Journal of research and development 3, 3, ст. 210–229, 1959.
6. Müller, M: Computer Go. In: Artif. Intell. 134, ст. 145–179, 2002.
7. DeepMind: StarCraft II ’mini games’ for AI research. 2017
8. Andersen: Towards a Deep Reinforcement Learning Approach for Tower Line Wars 10630, ст. 101–114, 2017.
9. What is Heuristics? Definition, Working, and Examples [Електронний ресурс] – Режим доступу до ресурсу: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-heuristics/.
10. Heuristic algorithms [Електронний ресурс] – Режим доступу до ресурсу: https://optimization.cbe.cornell.edu/index.php?title=Heuristic_algorithms.
11. What Is the Genetic Algorithm? [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html.

12. Introduction to Genetic Algorithms [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3
13. Genetic Algorithms [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/genetic-algorithms/.
14. What Is Q-Learning: The Best Guide To Understand Q-Learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning.
15. Q-Learning Algorithm: From Explanation to Implementation [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/q-learning-algorithm-from-explanation-to-implementation-cdbeda2ea187.
16. Тетріс [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%82%D1%80%D1%96%D1%81.
17. Intelligent agents in games [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0065245819300312.