Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Використання еволюціонуючої нейронної мережі в action-іграх

Виконав:

студент 2 курсу, групи КІТм-21-2
Полянська Є.О.
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп’ютерні інтелектуальні технології
Керівник проф. Корабльов М.М.

Допускається до захисту _____
(підпис)

Зав. кафедри проф. Руденко О.Г.
(підпис)

2022 р.

Харківський національний університет радіоелектроніки

Факультет	Комп'ютерної інженерії та управління
Кафедра	Комп'ютерних інтелектуальних технологій та систем
Рівень вищої освіти	другий (магістерський)
Спеціальність	123 Комп'ютерна інженерія
Тип програми	освітньо-професійна
Освітня програма	Комп'ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Полянській Єлизаветі Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Використання еволюціонуючої нейронної мережі в action-іграх

затверджена наказом по університету від " 7 " листопада 2022 р. № 1531

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2022 р.

3. Вхідні дані до роботи

1) аналіз нейронних мереж та роботи нейроеволюції для застосування в action-ігрі;
2) побудова моделі нейронної мережі для використання її в алгоритмі NEAT;
3) середовище розробки - PyCharm;
5) мова програмування – Python;
6) бібліотека Pygame.

4. Перелік питань, що потрібно опрацювати в роботі

1) огляд предметної області;
2) аналіз предмету дослідження;
3) дослідження нейронних мереж:
4) дослідження нейроеволюції
5) дослідження роботи алгоритму нейроеволюції наростаючих топологій;
6) розробка action-ігри з алгоритмом NEAT;
7) експериментальні дослідження;
8) висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Плакати - 18 арк. ф. А4

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача та узгодження теми проекту	07.11.2022	виканоно
2	Огляд стану проблеми та постановка задачі	01.09-14.09	Виканоно
3	Аналіз літератури за напрямком магістерської	14.09-21.09	Виканоно
	роботи		
4	Аналіз роботи нейроеволюції	21.09-09.10	Виканоно
5	Аналіз нейронних мереж	21.09-09.10	Виканоно
6	Аналіз роботи алгоритмі нейроеволюції	21.09-09.10	Виканоно
	наростаючих технологій		
7	Експериментальні дослідження	09.10-02.11	Виканоно
8	Підготовка графічного матеріалу	32.11-07.12	Виканоно
9	Перевірка виконаного проекту керівником	10.12.2022	виканоно
10	Захист проекту		

Дата видачі завдання 8 листопада 2022 р.

Студент _____

Керівник _____ проф. Корабльов М.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 73 с., 25 рис., 1 дод., 21 джерело.

НЕЙРОННІ МЕРЕЖІ, НЕЙРОЕВОЛЮЦІЯ, ЕВОЛЮЦІЯ, АЛГОРИТМ НЕЙРОЕВОЛЮЦІЇ НАРОСТАЮЧИХ ТОПОЛОГІЙ, ШТУЧНИЙ ІНТЕЛЕКТ, ACTION-ГРА, НАВЧАННЯ, ФУНКЦІЇ АКТИВАЦІЇ

У роботі пропонується створити action-гру з використанням алгоритму нейроеволюції наростаючих топологій для того, щоб на власні очі побачили навчання штучного інтелекту.

Метою кваліфікаційної роботи є дослідження на практиці роботи нейроеволюції та навчання штучного інтелекту за допомогою action-гри з використанням алгоритму нейроеволюції наростаючих топологій.

Об’єктом дослідження є action-гра написана з використанням алгоритму нейроеволюції наростаючих топологій….

Предметом дослідження є нейроеволюція та навчання штучного інтелекту грати в створену action-гру, що написана з використанням алгоритму нейроеволюції наростаючих топологій.

Одержані результати, а саме навчений штучний інтелект, який має надлюдську реакцію і вміє аналізувати ситуацію на високій швидкості, можна згодом використати в автопілоті для автомобілів.

ABSTRACT

Master`s thesis: 73 pages, 25 figures, 1 appendices, 21 sources.

EVOLUTION, NEURON NETWORKS, NEURO-EVOLUTION, , NEURO-EVOLUTION ALGORITHM OF GROWING TOPOLOGIES, ARTIFICIAL INTELLIGENCE, ACTION-GAME, LEARNING, ACTIVATION FUNCTIONS

The paper proposes to create an action game using the NeuroEvolution of Augmenting Topology in order to see with your own eyes the training of artificial intelligence.

The major goal of this thesis is a practical study of the work of neuroevolution and training of artificial intelligence with the help of action-game using the NeuroEvolution of Augmenting Topology.

The object of the research is an action game written using the NeuroEvolution of Augmenting Topology

The subject of research is neuroevolution and training artificial intelligence to play in the created action-game, which is written using the NeuroEvolution of Augmenting Topology

The results obtained, namely a trained artificial intelligence that has superhuman reactions and can analyze at high speed, can later be used in autopilot for cars.

The major goal of this thesis is…

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

АНОТАЦІЯ

КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Використання еволюціонуючої нейронної мережі в action-іграх

Виконав:
студент 2 курсу, групи КІТм-21-2
Полянська Є.О.
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп’ютерні інтелектуальні технології
Керівник проф. Корабльов М.М.

2022 р.

АНОТАЦІЯ

Полянська Є.О.. Використання еволюціонуючої нейронної мережі в action-іграх – Магістерська кваліфікаційна робота.

Актуальність теми дослідження. Поки існування цивілізації людина завжди буде розвиватися і навряд чи зупиниться, адже тоді людство скотиться до кам’яного віку. Століттями людина використовувала свій мозок та всі навики для того, щоб покращити собі життя та спрощувати поставлені задачі, тому ми створили ваги, калькулятор, комп’ютер, але в нині всього цього нам замало Тут вже необхідно дещо більше звичайного калькулятора, дещо схоже на людський інтелект. Тому на прикладі роботи біологічного мозку прийшла ідея створити штучний мозок. Відеоігри розвивають такі сфери як програмування, психологія, маркетинг, математика, дизайн тощо. Дослідження у сфері ШІ ведуться шляхом вивчення розумових здібностей людини та перекладу отриманих результатів у полі діяльності комп'ютерів. Таким чином, штучний інтелект отримує інформацію з різних джерел і дисциплін. Це інформатика, математика, лінгвістика, психологія, біологія, машинобудування.

Штучний інтелект вже застосовують для вирішення низки завдань розробки відеоігор. Насамперед, алгоритми штучного інтелекту дозволяють суттєво підвищити якість графіки та природність динаміки різних об'єктів: людей, транспорту, тварин, погодних проявів. Машинне навчання дозволяє виявляти найбільш релевантні інтегральні показники, відповідальні за моделювання цих процесів, що призводить до появи в нових іграх вкрай реалістичної графіки. І людство рухається далі, що якщо не нам навчати нейронну мережу, а вона сама буде навчатись.

Нейроеволюція досить цікава форма машинного навчання. Використовуючи алгоритми еволюції відбувається тренування нейронної мережі. Нейронна мережа

імітує роботу біологічних нервових систем. Данна форма машинного навчання зараз активно розвивається та застосовується в багатьох сферах.

Метою даної роботи є дослідження на практиці роботи нейроеволюції та навчання штучного інтелекту за допомогою action-гри з використанням алгоритму нейроеволюції наростаючих топологій.

Об’єктом дослідження є action-гра написана з використанням алгоритму нейроеволюції наростаючих топологій.

Предметом дослідження є нейронні мережі, схожість штучного мозку з біологічним, нейроеволюція та навчання штучного інтелекту грати в створену action-гру, що написана з використанням алгоритму нейроеволюції наростаючих топологій.

Методи дослідження для вирішення поставлених задач базуються на аналізі нейронних мереж, функцій активації, методів навчання, методів побудови нейронних мереж та будови самого нейрона, порівнянні біологічного мозку зі штучним.

У першому розділі розглянуто основні відомості про нейроеволюцію та action-ігри та були поставлені задачі дослідження. Action-ігри – жанр комп'ютерних ігор, в якому основна увага приділяється фізичним можливостям гравця, таким як: координації очей і рук та швидкості реакції. В екшен-іграх зазвичай аватар, що керується людиною, повинен знайти вихід із рівня, зібрати предмети і уникнути перешкод. Такі ігри дуже динамічні і вимагають високої концентрації уваги і швидкої реакції на події, що відбуваються в грі. Перешкоди та ворожі атаки виснажують здоров'я та запас життів аватара. За відсутності в нього життів, гра закінчується. В іншому випадку, коли серію рівнів успішно пройдено, гравець перемагає. Відеоігри розвивають такі сфери: програмування, психологія, маркетинг, математика, дизайн тощо. Дослідження у сфері ШІ ведуться шляхом вивчення розумових здібностей людини та перекладу отриманих результатів у полі діяльності комп'ютерів. ШІ застосовують в інформатиці, математиці, лінгвістиці, психології, біології, машинобудуванні. Комп'ютери намагаються імітувати інтелект людини. Штучний інтеллект вже застосовують для розробки відеоігор. Насамперед,

алгоритми штучного інтелекту дозволяють суттєво підвищити якість графіки та природність динаміки різних об'єктів: людей, транспорту, тварин, погодних проявів. Нейроеволюція – форма машинного навчання, яка використовує еволюційні алгоритми для тренування нейромережі. Нейроеволюційні алгоритми (НЕА) допомогають при вирішенні таких проблем:

- скорочення простору пошуку,
- підвищення швидкості навчання,
- підвищення якості одержуваних рішень.

У другому розділі були досліджені будова нейрона, нейнні мережі їх типи та моделі, функції активації та різні типи навчання. Перед традиційними алгоритмами можливість навчання - одна з головних переваг нейронних мереж. Технічно навчання полягає у знаходженні коефіцієнтів зв'язків між нейронами. У процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними та вихідними, а також виконувати узагальнення. Це означає, що у разі успішного навчання мережа зможе повернути правильний результат на підставі даних, які були відсутні у навчальній вибірці, а також неповних та «зашумлених», частково спотворених даних. Нейронна мережа використовує спрощену модель обробки інформації людським мозком. Нейромережі працюють, обраховуючи велику кількість пов'язаних між собою оброблюваних елементів, які представляють абстрактну версію нейронів. Були розглянуті такі типи нейронних мереж:

- нейронні мережі з прямим зв’зком;
- персептрон;
- довга короткострокова пам’ять;
- мережа радіально-базисних функцій.

Проаналізовані методи навчання нейронної мережі, а саме:

- навчання з учителем;
- навчання без учиталя;
- навчання з частковим залученням вчителя;
- навчання з підкріпленням.

На основі аналізу, зазначеного вище, зроблений вібір нейронної мережі, функції

активації та методу навчання

У третьому розділі пропонується детальний опис нашої action-гри під назвою realUAcat, розбір роботи алгоритму NEAT. Була застосована середа розробки PyCharm та бібліотека Pygame. Створена гра жанру action, де роль головного героя відіграє котик, що стрибає. Головна задача, що стоїть перед ним – це перестрибувати через перепони, тобто квіти, які мають різну ширину. По суті котик біжить на місті, а земля та квіти рухаються йому на зустріч, із часом гра пришвидшується. При зіткненні з перепоною, котик гине і гра починається наново. Алгоритм нейроеволюції наростаючих топологій розвиває нейронну мережу. На вході алгоритм приймає початкові значення вузлів, саме для цього ми і вибираємо просту нейронну мережу, в процесі навчання випадковим чином додає нові вузли і утворює додаткові зв'язки між ними. Особливістю NEAT є те, що він починає роботу зі спрощеною моделлю і ускладнює її лише у разі потреби. Залежно від кількості еволюцій, заданих користувачем, алгоритм еволюціонує, і кожне покоління осіб є результатом відтворення та мутації найкращих особин попереднього покоління

НЕЙРОННІ МЕРЕЖІ, НЕЙРОЕВОЛЮЦІЯ, ЕВОЛЮЦІЯ, АЛГОРИТМ НЕЙРОЕВОЛЮЦІЇ НАРОСТАЮЧИХ ТОПОЛОГІЙ, ШТУЧНИЙ ІНТЕЛЕКТ, ACTION-ГРА, НАВЧАННЯ, ФУНКЦІЇ АКТИВАЦІЇ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МРТ – магнітно-резонансна томографія

НМ – нейронна мережа

НЕА – нейроеволюційні алгоритми

ШІ – штучний інтелект

FPS – кадри в секунду (англ. Frames Per Second)

GAN – генеративно-змагальна мережа (англ. Generative Adversarial Network,)

IDE – интегроване середовище разробки (англ. Integrated Development Environment)

LSTM – довга короткострокова пам'ять (англ. Long Short-Term Memory)

MLP – багатошаровий перцептрон (англ. Multilayered Perceptron)

МОВА – багатокористувацька онлайнова бойова арена (англ. Multiplayer Online Battle Arena)

NEAT – алгоритм нейроеволюціх наростаючих технологій (англ. NeuroEvolution of Augmenting Topology)

RBF –радіальна базова функція (англ. Radial basis function)

RNN – рекурентна нейронна мережа (англ. Recurrent Neural Network)

ВСТУП

Впродовж усього існування цивілізації людина завжди розвивалася і навряд чи зупинимось. Людина спрощувала і покращувала своє життя століттями, створили ваги, калькулятор, комп’ютер, але в XXI столітті всього цього вже нам не достатньо, хоча і значно полегшує нам життя. Зараз нам хочеться автоматизувати складні задачі такі як розпізнавання обличь, відокремлення хворих клітин від здорових та ігри. Тут вже необхідно дещо більше звичайного калькулятора, дещо схоже на людський інтелект.

Комп’ютер має більш жорстку логіку, що складається з нулів та одиниць і ніколи не зможе змагатися з гнучкістю біологічного мозку. Тому прийшла ідея спробувати створити штучний мозок на прикладі роботи біологічного. Реального мозку з нейронами замість логічних алгоритмів, який наділений здатністю робити висновки. Так з’явилась теорія нейронних мереж, що стала найпотужнішим і найкориснішим підходом в розробці штучного інтелекту. Нині штучний інтелект вже допомагає розпізнавати через камеру обличчя людей на вулиці, автоматичне розпізнавання номерів автомобілів та навіть обіграти в шахи гросмейстера світового рівня.

Нейронні мережі мають нейрони і зв’язки між ними, що забезпечують перетворення вхідних сигналів на значний вихідний результат. В області машинного навчання ці мережі часто ініціалізуються з випадковими зв'язками між нейронами, після чого мережа навчається, поки не почне поводитися належним чином. Такий підхід цілком може бути застосований, однак у тварин існує безліч простих нервових систем, що працюють від народження: ніхто не вчить рибу плавати або метеликів літати, незважаючи на те, що їхня поведінка створюється мережами нейронів. Їхні нервові системи є результатом не випадкової ініціалізації та подальшого навчання, а еволюції. Через безліч поколінь природа створила таку схему з клітин та зв'язків, яка забезпечує складну та успішну поведінку. Для створення нейронних мереж, які забезпечують поведінку без навчання, можна

використовувати нейроеволюцію. Протягом тривалого часу еволюційні алгоритми піддають генетичнй код еволюції. Протягом багатьох поколінь невеликі мутації збільшують сприятливу складність та додають функції, що стимулюють подальший розвиток та вдосконалення.

Нейроеволюція визнається перспективним напрямом вирішення проблемних питань машинного навчання нейромереж в областях, де реалізація навчання за прецедентами практично неможлива: робототехніка, багатоагентні системи, при пошуку оптимальних рішень в ігрових завданнях, у прогнозуванні тощо.

Метою магістерської кваліфікаційної роботи є дослідження на практиці роботи нейроеволюції та навчання штучного інтелекту за допомогою action-гри з використанням алгоритму нейроеволюції наростаючих топологій.

1 ОСНОВНІ ВІДОМОСТІ ПРО НЕЙРОЕВОЛЮЦІЮ ТА ACTION-ІГРИ

1.1 Нейроеволюція – майбутнє людства

Нейроеволюція досить цікава форма машинного навчання. Використовуючи алгоритми еволюції відбувається тренування нейронної мережі. Нейронна мережа імітує роботу біологічних нервових систем, що дозволяє відійти від комп’ютерного мислення коли маємо лише 1 або 0. Нейронна мережа дозволяє зробити крок вперед в нашому розвитку і створити дещо схоже на людський мозок. У головному мозку людини, що є найрозвиненішим, нараховується близько 100 мільярдів нейронів. Нейроеволюція захоплює, це дуже цікаво спостерігати як вона сама навчається та вдосконалюється. Данна форма машинного навчання зараз активно розвивається та застосовується в багатьох сферах.

Для того, щоб на власні очі проспостерігати даний процес, а саме навчання нейронної мережі, можна створити звичайну action-гру, де буде відбуватись еволюція головного персонажа поки нейронна мережа не навчиться.

1.1 Action-ігри

Action – жанр комп'ютерних ігор, в якому основна увага приділяється фізичним можливостям гравця, у тому числі координації очей і рук і швидкості реакції.[1] Жанр представлений у безлічі різновидів від файтингів, шутерів та платформерів, які вважаються найбільш важливими для жанру, до MOBA (Multiplayer Online Battle Arena) та деяких стратегій у реальному часі, які можна віднести до жанру екшен. На рисунку 1.1 наведений приклад action-гри у жанрі платформер.

Рисунок 1.1 – Приклад action-гри у жанрі платформер

В екшен-іграх зазвичай гравець керує протагоністом або аватаром. Цей персонаж повинен знайти вихід із рівня, зібрати предмети і уникнути перешкод. Дія таких ігор розвивається дуже динамічно і вимагає високої концентрації уваги і швидкої реакції на події, що відбуваються в грі. Перешкоди та ворожі атаки виснажують здоров'я та запас життів аватара. За відсутності в нього життів, гравець отримує повідомлення "Game over". В іншому випадку, коли серію рівнів успішно пройдено, гравець перемагає.

Тим не менш, у ряді ігор, особливо в аркадах, кількість рівнів може бути нескінченною, і перемогти в таких іграх неможливо. У цьому випадку метою гравця стає отримання якомога більшої кількості очок, збираючи предмети та знищуючи ворогів. До екшен-ігор може бути віднесена будь-яка гра, де перемога над супротивником забезпечується завдяки фізичній перевазі, наприклад, найкращим прицілюванням або меншим часом реакції.

Поява 1978 року такого блокбастера, як «Space Invaders», викликало поворот у промисловості комп'ютерних ігор у бік активних игр.[1]

Слідом за «Space Invaders», з'явилися такі популярні ігри, як Asteroids 1979, Pac-Man 1980-го, що представлені на рисунку 1.2 а) і б) відповідно, і безліч інших аркадних відеоігор, які задали канони жанру. Paperboy, що вийшла в 1984, змогла перетворити доставку газет в ігровий процес, демонструючи тим самим універсальність цього жанру. Robotron: 2084, що вийшов 1982 року, започаткував шутери.[1]

а)

б)

Рисунок 1.2 – Перші action-ігри: а) Asteroids 1979; б) Pac-Man 1980-го

1.3 Роль штучного інтелекту в іграх

Ігри як комп’ютерні так і мобільні нині займають значну частину в багатьох сферах нашого життя. Це не тільки розваги, як можна подумати на перший погляд, це перш за все розвиток технологій. За останні два десятиліття ця сфера кардинально змінилась. Від простих ігор з примітивною графікою ми дійшли до

максимального реалізму, від 2Д до 3Д, нині ми маємо неймовірні, різноманітні світи. Відеоігри розвивають такі сфери як програмування, психологія, маркетинг, математика, дизайн тощо.

Дослідження у сфері ШІ ведуться шляхом вивчення розумових здібностей людини та перекладу отриманих результатів у полі діяльності комп'ютерів. Таким чином, штучний інтелект отримує інформацію з різних джерел і дисциплін. Це інформатика, математика, лінгвістика, психологія, біологія, машинобудування. За підсумками масиву даних з допомогою технології машинного навчання комп'ютери намагаються імітувати інтелект людини.[2]

Штучний інтелект вже застосовують для вирішення низки завдань розробки відеоігор. Насамперед, алгоритми штучного інтелекту дозволяють суттєво підвищити якість графіки та природність динаміки різних об'єктів: людей, транспорту, тварин, погодних проявів. Машинне навчання дозволяє виявляти найбільш релевантні інтегральні показники, відповідальні за моделювання цих процесів, що призводить до появи в нових іграх вкрай реалістичної графіки.

Також, всі геймери мріють, щоб алгоритми, що відповідають за моделювання дій суперника у грі, стали хоч трохи наближеними до реальних. Машинне навчання дозволяє анімувати ігрових суперників, зробивши гру набагато живішою та залучаючою. Використання машинного навчання дозволить реалізувати у низці ігор таку довгоочікувану можливість, як варіація сюжетних ліній.

1.4 Нейроеволюція – форма машинного навчання

Нейроеволюція – форма машинного навчання, яка використовує еволюційні алгоритми для тренування нейромережі. Цей підхід використовується в таких галузях як ігри та керування приводами роботів. У цих випадках досить просто виміряти продуктивність нейромережі, тоді як реалізувати навчання з учителем дуже важко чи практично неможливо[3]. Цей метод навчання належить до категорії методів навчання із підкріпленням.

Навчання з підкріпленням є проміжним типом завдань між навчанням з учителем по прецедентам (регресійний аналіз, класифікація) та навчанням без вчителя, де потрібно знайти закономірності даних (кластеризація) – у процесі навчання враховується взаємодія із середовищем.

Нейроеволюційні алгоритми (НЕА) націлені на вирішення таких проблем:

- скорочення простору пошуку,
- підвищення швидкості навчання,
- підвищення якості одержуваних рішень[4].

Евристики, що застосовуються в нейроеволюційних алгоритмах, поділяються на такі категорії:

- обмеження на структуру нейромереж,
- вибір операторів зміни нейромереж,
- способи використання еволюційних операторів,
- евристики для цільової функції.

Залежно від розв'язуваних завдань розрізняють три типи нейроеволюційних алгоритмів:

- алгоритми пошуку значень ваги сполук нейромережі при фіксованій структурі;
- алгоритми налаштування структури нейромережі;
- алгоритми налаштування параметрів функцій активації нейронів;
- алгоритми, що становлять різні комбінації типів НЕА[4].

1.4.1 Нейроеволюційні алгоритми

Нейроеволюційний алгоритм починає свою роботу з нейромереж без нейронів прихованого шару. Синтез нейромережі йде у напрямку поступового ускладнення її топології. Генотип у нейроеволюційного алгоритму є безліччю суміжних нейронів і безліччю синаптичних ваг їх сполук. Кожен нейрон кодується його номером у поколінні популяції та типом шару (вхідний, прихований, вихідний).

Безліч синапсів кодується покажчиками на суміжні нейрони, значення синаптичних ваг, прапор, що вказує використовується чи ні з'єднання, а також на «історію» з'єднання в поколіннях популяції[3].

Для генерації нейромереж не використовується оператор кросинговера, оскільки його застосування носить деструктивний характер, а нащадки нейромереж, як правило, має функцію придатності гірше, ніж у батьківських хромосом. Населення розглядається як центральний об'єкт НЕА, а еволюція нейромережі є процесом її пристосування на поведінковому рівні[3]. Цей рівень абстракції не передбачає рекомбінації, тому оператор кросинговера відсутній.

1.5 Нейроеволюція в action-грі

Людині завжди хочеться швидше, вище та краще, але нажаль важко «стрибнути вище голови», все залежить від наших фізичних можливостей. Штучний інтелект дозволить нам це зробити. Нехай гра навчиться грати сама в себе, нейроеволюція допоможе перестрибнути бар’єр людських можливостей.

З проходженням кожного покоління нейромережа буде самостійно навчатись і виводити найкращу еволюцію головного героя. Сама гра представляє собою персонажа, що біжить та перестрибує через перепони, і поступово швидкість гри збільшується. В межах одного покоління на старті маємо деяку кількість головних героїв, під час навчання нейромережі відсіються найменш здатні і в кінці ми отримаємо переможця. В наступномі поколінні відбувається все теж саме, але персонажі отримують генетику першого покоління і так далі, до безкінечності. Щоб легше було визначати переможця та навчатись самій нейромережі буде система оцінювання.

Для розробки нейроеволюційної гри, яка буде грати сама в себе були розглянуті декілька існуючих рішень, наприклад гра MarI/O від SethBling або нейроеволюція кіберкальмарів.[3] Після тестування даних програм були виявлені переваги та недоліки, проведений порівняльний аналіз та зроблені висновки.

1.6. Постановка задачі дослідження

Метою магістерської кваліфікаційної роботи є дослідження на практиці роботи нейроеволюції та навчання штучного інтелекту за допомогою action-гри з використанням алгоритму нейроеволюції наростаючих топологій.

Для досягнення мети дослідження потрібно вирішити такі завдання, як:

розглянути будову НМ;

проаналізувати архітектури НМ;

розглянути підходи до нейроеволюції;

розглянути роботу алгоритму нейроеволюції наростаючих топологій;

створити гру з використанням алгоритму нейроеволюції наростаючих топологій;

експериментуючи дізнатись як швидко штучний інтелект навчиться грати.

Вирішення цих завдань дозволить створити нейроеволюціонуючу action-гру з використанням алгоритму нейроеволюції наростаючих топологій, в яку навчиться грати штучний інтелект. Який після навчання можна буде використовувати, наприклад, як автопілот для автомобілів.

2 АНАЛІЗ НЕЙРОННИХ МЕРЕЖ, ЇХ ТИПІВ ТА МОДЕЛЕЙ ДЛЯ ВИКОРИСТАННЯ В ACTION-ІГРАХ

2.1 Штучні нейронні мережі, як прототип біологічного мозку

Нейронні мережі – один з найкорисніших підходів розробки штучного інтелекту. Нейронні мережі, або, точніше, штучні нейронні мережі, представляє собою технологію, що сягає корінням в безліч дисциплін: нейрофізіологія, математику, статистику, фізику, комп'ютерні науки і техніку. Вони знаходять своє застосування в таких різнорідних областях, як моделювання, аналіз часових рядів, розпізнавання образів, обробка сигналів і управління завдяки одній важливій властивості-здатності навчатися на основі даних за участю вчителя або без його втручання.

Мозок тварин ставив вчених у глухий кут, оскільки навіть у таких малих представників живої природи, як птахи, він демонструє незрівнянно більші здібності, ніж цифрові комп'ютери з величезною кількістю електронних обчислювальних елементів та пам'яттю неймовірних обсягів, що працюють на частотах, недосяжних для живого мозку із плоті та крові.[5] Тоді зосередили увагу на архітектурних відмінностях. У традиційних комп'ютерах дані обробляються послідовно, за чітко встановленими правилами. У їх холодних запрограмованих розрахунках немає місця неоднозначності та неясності. З іншого боку, поступово ставало зрозуміло, що мозок тварин, незважаючи на здається уповільненість його робочих ритмів у порівнянні з комп'ютерами, обробляє сигнали паралельно і що невизначеність є істотною відмінністю його діяльності. Основною одиницею біологічного мозку є нейрон. Рисунок 2.1 описує будову біологічного нейрона.

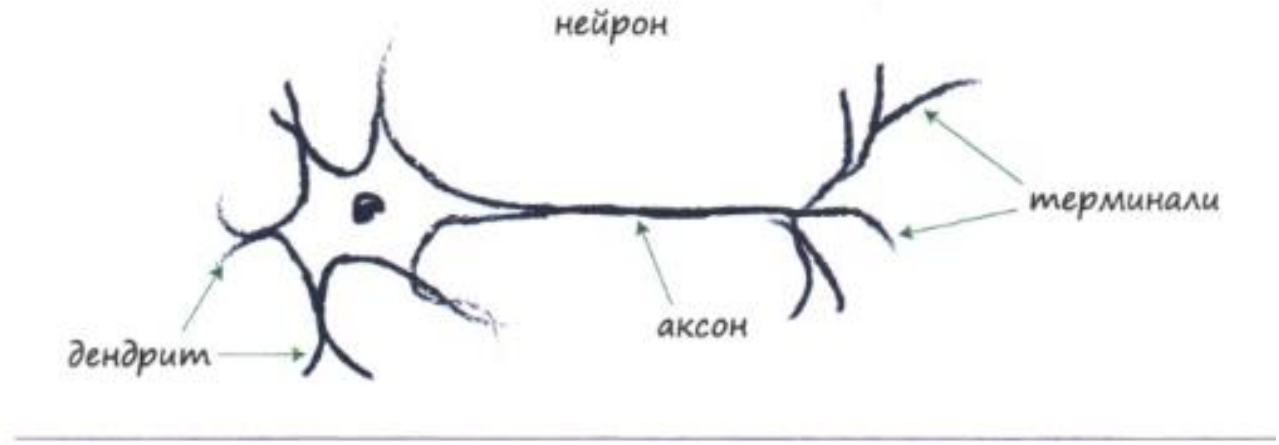


Рисунок 2.1 – Будова біологічного нейрона

Штучна нейронна мережа є системою з'єднаних і взаємодіючих між собою простих процесорів – штучних нейронів.[5] Такі процесори зазвичай досить прості порівняно з процесорами, які у персональних комп'ютерах. Кожен процесор подібної мережі має справу лише з сигналами, які він періодично отримує, та сигналами, які він періодично надсилає іншим процесорам. І, тим щонайменше, будучи з'єднаними у досить велику мережу з керованим взаємодією, такі окремі прості процесори разом здатні виконувати досить складні завдання.

Можливість навчання - одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає у знаходженні коефіцієнтів зв'язків між нейронами.[5] У процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними та вихідними, а також виконувати узагальнення. Це означає, що у разі успішного навчання мережа зможе повернути правильний результат на підставі даних, які були відсутні у навчальній вибірці, а також неповних та «зашумлених», частково спотворених даних.

Реальні біологічні нейрони мають кілька входів, а не лише один. Їх просто комбінують підсумовуючи відповідні значення, і результативна сума буде служити вхідним значенням для сигмоїди, що керує вихідним значенням. Така схема відображає принцип роботи нейронної мережі. Рисунок 2.2 ілюструє ідею комбінування вхідних значень і порівняння результатуючої суми з пороговим значенням

Рисунок 2.2 – Ідея комбінування вхідних значень і порівняння результатуючої суми з пороговим значенням

Якщо комбінований сигнал недостатньо сильний, то сигмоїда пригнічує вихідний сигнал. Якщо ж сума x досить велика, то функція збуджує нейрон. Навіть якщо тільки один сигнал досить сильний, у той час як усі інші мають невелику величину, то й цього може цілком вистачити для збудження нейрона.[5] Нейрон може збудитися і тоді, коли кожен із сигналів, взятих окремо, має недостатню величину, але, будучи взятими разом, вони забезпечують перевищення порога. У цьому вже відчувається прототип складних і в певному сенсі невизначених обчислень, на котрі здатні подібні нейрони.

Виділяють кілька базових типів завдань, на вирішення яких можна використовувати нейромережі[5]:

- класифікація. Для розпізнавання облич, емоцій, типів об'єктів: наприклад, квадратів, кіл, трикутників. Також для розпізнавання образів, тобто вибору конкретного об'єкта із запропонованої множини: вибір квадрата серед трикутників.
- регресія. Для визначення віку за фотографією, складання прогнозу біржових курсів, оцінки вартості майна та інших завдань, що вимагають отримання в результаті обробки конкретного числа.
- прогнозування часових рядів. Для складання довгострокових прогнозів з

урахуванням динамічної часової низки значень. Наприклад, нейромережі застосовуються для прогнозування цін, фізичних явищ, обсягу споживання та інших показників. По суті навіть роботу автопілота Tesla можна віднести до процесу прогнозування часових рядів.

- кластеризація. Для вивчення та сортування великого обсягу нерозмічених даних за умов, коли невідомо кількість класів на виході, тобто для об'єднання даних за ознаками. Наприклад, кластеризація застосовується виявлення класів картинок і сегментації клієнтів.

- генерація. Для автоматизованого створення контенту чи його трансформації. Генерація за допомогою нейромереж застосовується для створення унікальних текстів, аудіофайлів, відео, розфарбовування чорно-білих фільмів та навіть зміни навколишнього середовища на фото.

2.2 Модель нейрона

В основі штучних НМ лежить нейрон. Нейрон – це одиниця обробки інформації в нейронній мережі. На рисунку 2.3 показана нелінійна модель нейрона.

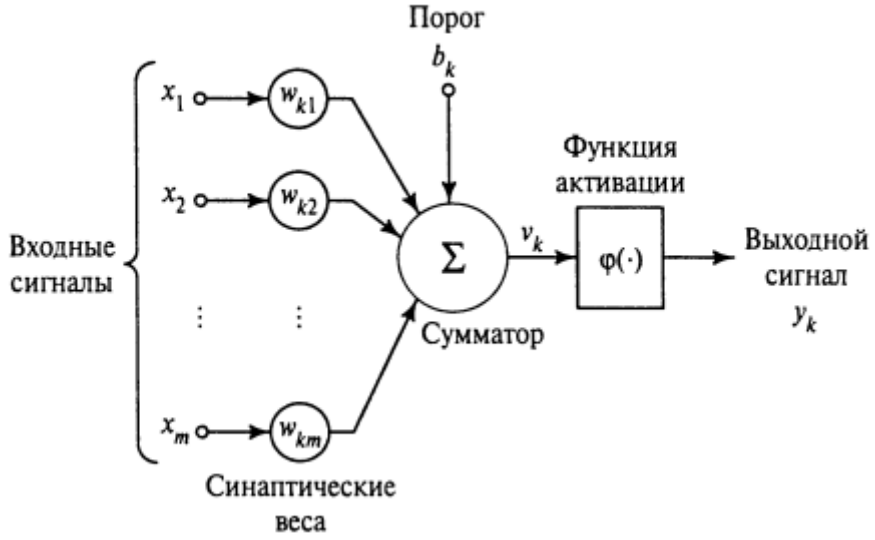


Рисунок 2.3 Нелінійна модель нейрона

Можна виділити три основних елементи, це синапси, суматор та функція активації.

Набір синапсів, де кожен характеризується своєю вагою або силою. Зокрема сигнал x_j на вході синапса j, пов'язаного з нейроном k, помножується на вагу w_{kj}.[6] Перший індекс відноситься до нейрона, що розглядається, а другий до вхідного закінчення синапсу з котрим пов'язана дана вага. Вага може бути збудливою, або гальмуючою, тобто позитивним або ж негативним значенням. Варто зазначити, що на відміну від синапсів мозку синаптичні ваги штучного нейрона можуть мати як позитивне так і негативне значення.

Суматор сумує вхідні сигнали, зважені щодо відповідних синапсів нейрона. Цю операцію можна описати як лінійну комбінацію.

Функція активації обмежує амплітуду вихідного сигналу нейрона. Ця функція також називається функцією стиснення. Зазвичай нормалізований діапазон амплітуд виходу нейрона лежить в інтервалі [0, 1] або [-1,1].[6]

В моделі нейрона, що зображено на рисунку 2.3 є пороговий елемент, зазначений символом b_k. Дана величина відображає збільшення чи зменшення вхідного сигналу, що подається на функцію активації. В математичному вигляді, для k-го нейрона можна представити таку функцію, де Y_k представляє рівень активації нейрона (формула 1.1)[6]

$$Y_k = \sum_j w_{kj} x_j \qquad (2.1)$$

2.3 Типи функції активації

Функції активації визначають вихідний сигнал нейрона. Розглянемо, які бувають функцій активації.

Перший, сигмоїдальна функція. Це гладка монотонно зростаюча нелінійна функція, її графік нагадує букву S (рисунок 2.4). Певно, є найрозповсюдженішою функцією, що використовують для створення штучних нейронних мереж. Вона дозволяє нейронам як підвищувати слабкі сигнали так і не насищатись від сильних

сигналів. Не є бінарною, що дозволяє зробити активацію аналоговою. Прикладом сигмоїдальної функції може бути логістична функція, яка має вигляд:[7]

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.2}$$

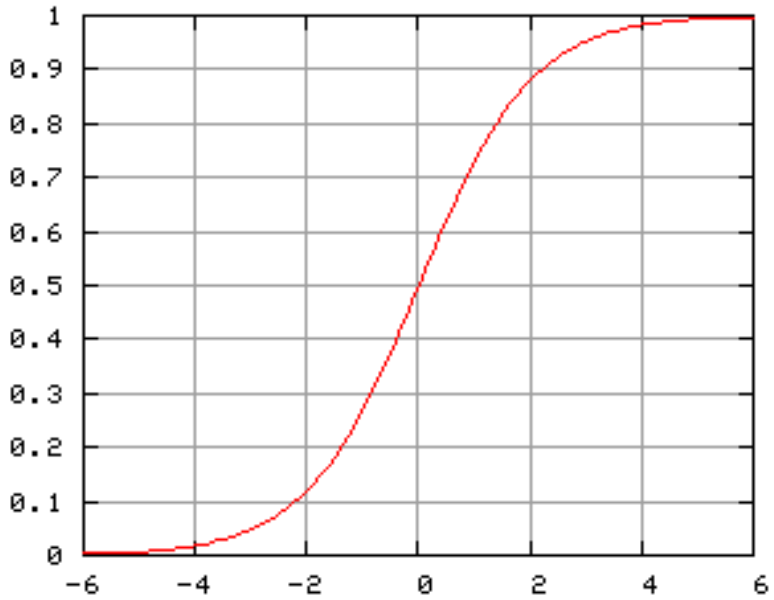


Рисунок 2.4 – Сигмоїдальна функція (логістична крива)

Другий, функція Хевісайда, або функція одиничного стрибка. Дана функція дорівнює нулю для негативних значень аргументу та одиниці – для позитивних. На рисунку 2.5 показаний даний тип функції.

Ця функція описується наступним чином: [6]

$$\varphi(v) = \begin{cases} 1, v \geq 0; \\ 0, v < 0; \end{cases} \tag{2.3}$$

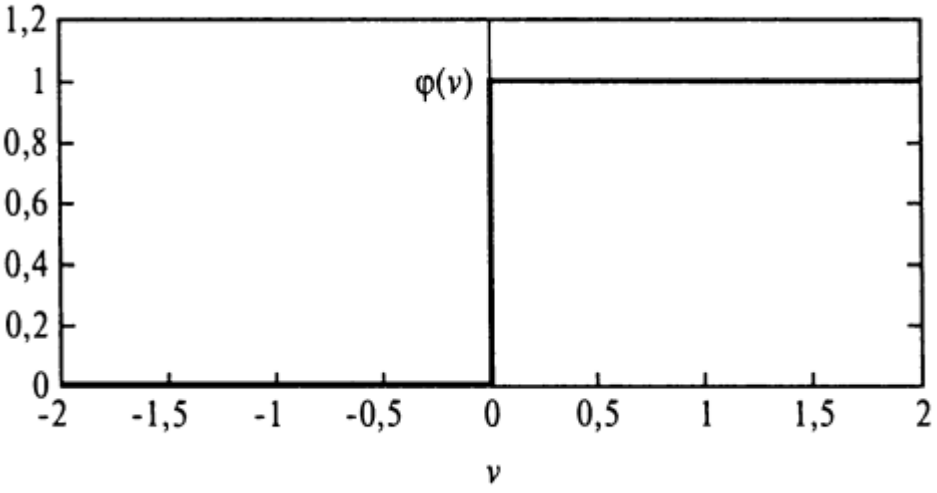


Рисунок 2.5 Функція Хевісайда (одиничного стрибка)

Шматково-лінійна функція зображена на рисунку 2.6. Функція, визначена на множині дійсних чисел, лінійна на кожному з інтервалів, що складають область визначення.

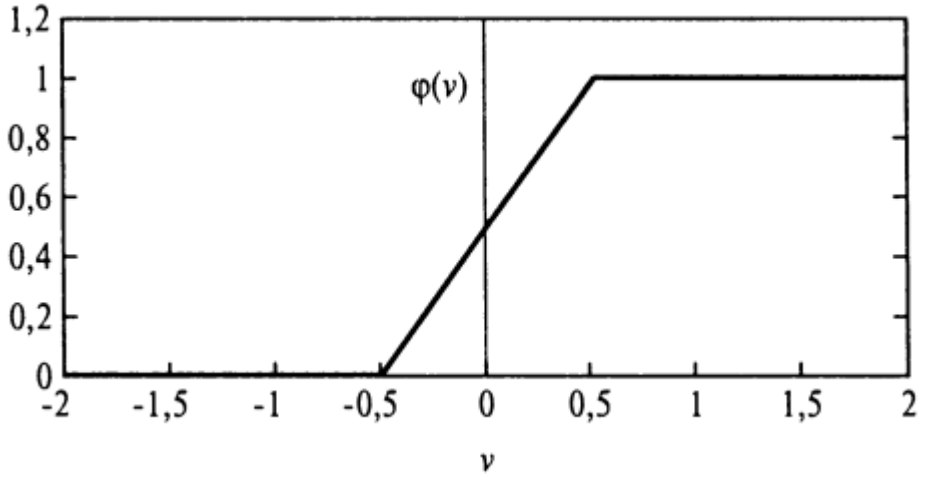


Рисунок 2.6. Шматково-лінійна функція

Шматково-лінійна функція описується наступним чином[6]:

$$\varphi(v) = \begin{cases} 1, & v \geq \frac{1}{2}; \\ |v|, \frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \quad (2.4)$$

де, коефіцієнт посилення у лінійній області передбачається рівним одиниці. Наступні два варіанти можна вважати особливою формою шматково-лінійної функції. Якщо лінійна область оператора не досягає порога насичення, він перетворюється на лінійний суматор.[7] Якщо коефіцієнт посилення лінійної області прийняти нескінченно більшим, то шматково-лінійна функція вироджується в граничну.

Досить часто застосовується форма сигмоїдної нелінійності - гіперболічний тангенс. На рисунку 2.7 можна побачити графік функції гіперболічного тангенсу. Вона описується наступним виразом[6]:

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1 \quad (2.5)$$

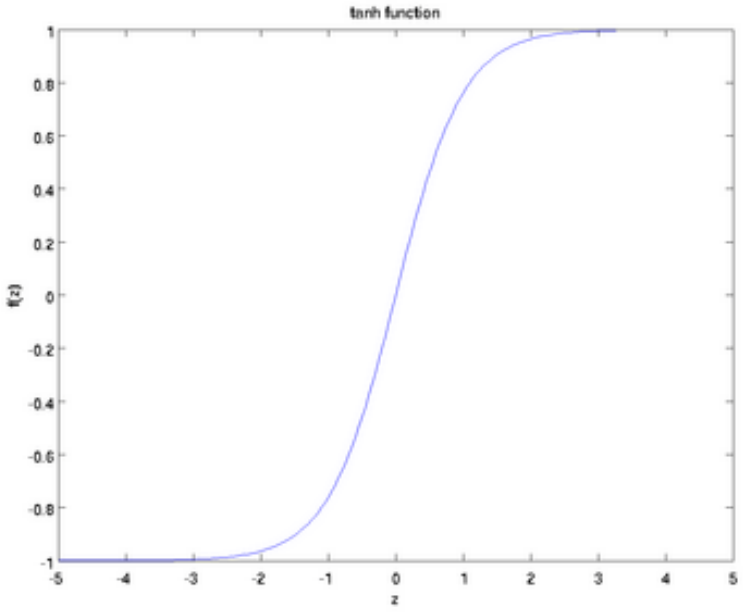


Рисунок 2.7 Функція гіперболічного тангенсу

Дана функція має ті ж самі характеристики що і у сигмоїди. Так як вона не лінійна, то добре підходить для комбінації шарів, діапазон значень функції (-1,1).. Отже активаційна функція не перевантажиться від великих значень. Градієнт же функції активації гіперболічного тангенсу більший ніж функції сигмоїда.

2.4 Загальна будова нейронної мережі

Нейронні мережі це спрощені моделі роботи нервової системи живих організмів. Базові блоки називаються нейронами і зазвичай згруповані шари, як показано на рисунку 2.8.

Нейронна мережа використовує спрощену модель обробки інформації людським мозком. Нейромережі працюють, обраховуючи велику кількість пов'язаних між собою оброблюваних елементів, які представляють абстрактну версію нейронів.

Обробні блоки, тобто нейрони – згруповані шари. У типовій нейромережі є три частини: нейрони вхідного шару являють собою вхідні поля, є один або кілька прихованих шарів і є вихідний шар, що містить один або кілька нейронів, що представляють поля призначення.[8] Кожному з'єднанню між нейронами призначається та чи інша сила впливу, чи вага. Вхідні дані надходять у перший шар, далі значення поширюються шарами від кожного нейрона даного шару в кожен нейрон наступного шару. Остаточний результат знімається з вихідного шару.

Нейронна мережа навчається шляхом перегляду записів; для кожного запису нейронна мережа генерує прогноз і, якщо прогноз неправильний, вносить поправки у ваги. Процес повторюється велику кількість разів, і точність передбачень поступово підвищується, доки не спрацьовує один із критеріїв зупинки.

Спочатку всі ваги випадкові та відповіді нейронної мережі на вхідні сигнали, швидше за все, безглузді. Нейронну мережу навчають. Приклади, яким відомі вихідні значення, багаторазово пред'являються нейронній мережі, і щоразу її відгук порівнюється з відомою відповіддю.[8] Інформація від такого порівняння

передається назад до нейронної мережі, поступово змінюючи ваги. У міру навчання нейронна мережа починає видавати відповіді, які дедалі точніше відтворюють відомі відповіді. Після навчання нейронну мережу застосовують до майбутніх спостережень, котрим результат невідомий.

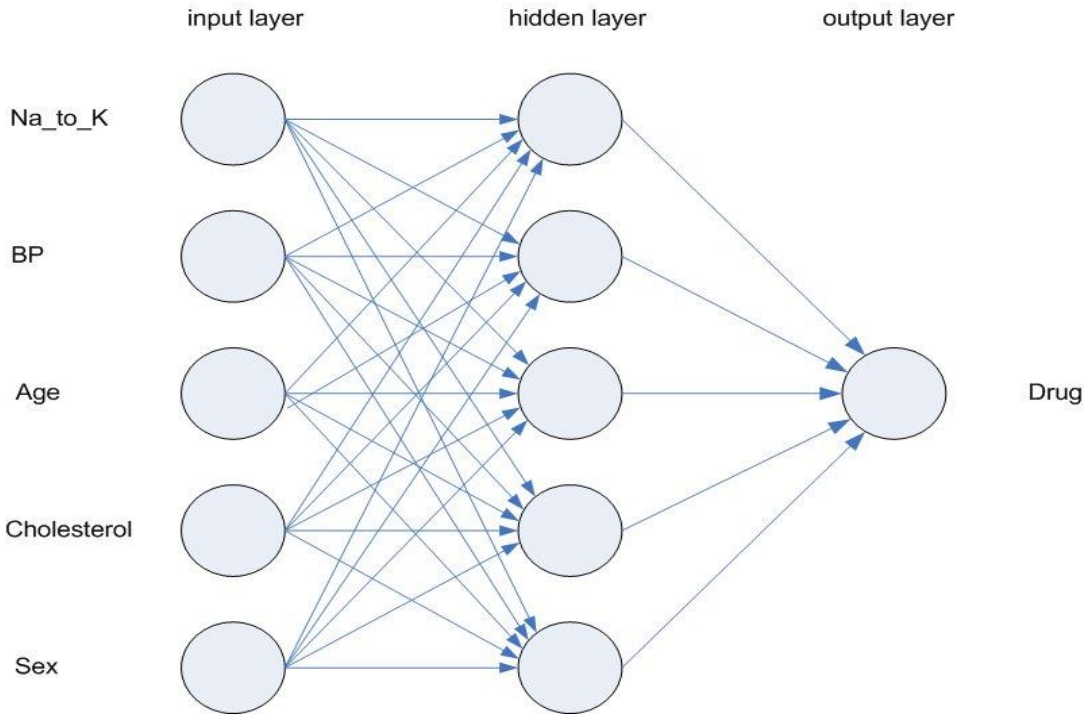


Рисунок 2.8 – Структура нейронної мережі

Тип моделі визначає спосіб з'єднання предикторів з полями призначення через один або кілька прихованих шарів нейронної мережі. Багатошаровий перцептрон (MLP) створює можливість складніших взаємозв'язків, можливо, ціною великих витрат часу навчання і оцінку.[8] Радіальна базова функція (RBF) порівняно з MLP може відрізнятися меншими витратами часу на навчання та оцінку, можливо, ціною меншої потужності передбачення.

Сховані шари. Приховані шари нейронної мережі містять обробні блоки, що не спостерігаються (нейрони). Значення кожного прихованого нейрона – це деяка функція предикторов;[8] Точна форма цієї функції частково залежить від типу мережі. Багатошаровий перцептрон може мати один або два приховані рівні; мережа радіальної базової функції може мати один прихований шар.

2.5 Структури мереж

Загалом можна виділити три архітектури нейронних мереж:

- одношарові мережі прямого розповсюдження;
- багатошарові мережі прямого розповсюдження;
- рекурентні мережі.

У багатошаровій нейронній мережі нейрони розташовуються шарам. У найпростішому випадку в мережі існує вхідний шар вузлів джерела, інформація від якого передається на вихідний шар нейронів, але не навпаки.[6] Така мережа називається мережею прямого поширення або ациклічною. Одношарова мережа прямого поширення з чотирьох вузлів в кожному з шарів, тобто на вхідному і вихідному зображена на рисунку 2.9. Називається вона так, бо при підрахуванні кількості шарів ми не беремо до уваги вхідні вузли, так як вони не виконують жодних обчислень.[6]

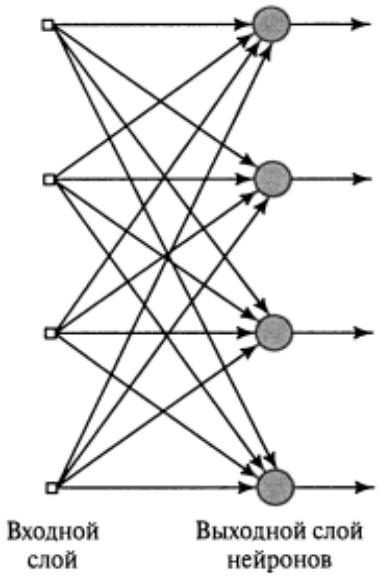


Рисунок 2.9 Одношарова мережа прямого поширення

Багатошарові мережі прямого поширення мають таку назву, бо включають в себе один, або декілька скритих шарів, вузли яких називають скритими нейронами. Сутність даних шарів полягає в тому, що вони виконують роль посередника між

вхідним сигналом і виходом нейронної мережі. Таким чином підвищується рівень взаємодії нейронів і за рахунок наявності додаткових синаптичних зв'язків мережа дозволяє виділяти статистики високого порядку та глобальні властивості даних. Коли розмір вхідного шару досить велика, скриті нейрони мають здатність виділяти статичні залежності високого порядку.[6]

Вузли джерела вхідного шару мережі формують відповідні елементи шаблону активації, які складають вхідний сигнал, що надходить на нейрони другого шару, тобто першого прихованого шару, аналогічно вихідні сигнали другого шару використовуються як вхідні для третього шару і т.д. Зазвичай нейрони кожного з шарів мережі використовують як вхідні сигнали, вихідні сигнали нейронів тільки попереднього шару.[6] Набір вихідних сигналів нейронів вихідного шару мережі визначає загальний відгук мережі, цей вхідний образ, сформований вузлами джерела вхідного шару. Мережа, показана на рисунку 2.10 називається мережею 10-4-2, так як вона має 10 вхідних, 4 приховані та 2 вихідні нейрони.

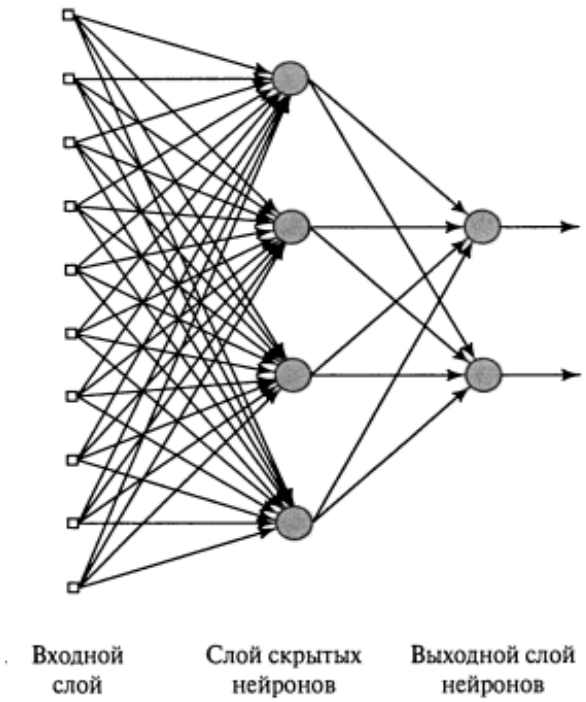


Рисунок 2.10 Багатошарова повнозв'язкова мережа прямого поширення з одним скритим слоєм.

Дана нейронна мережа вважається повнозв'язковою, тому що всі вузли кожного конкретного шару пов'язані з усіма вузлами суміжних шарів. Якщо деякі із синаптичних зв'язків відсутні, така мережа буде називатися неповнозв'язною.[6]

Рекурентна нейронна мережа (Recurrent Neural Network, RNN) відрізняється від мережі прямого поширення наявністю принаймні одного зворотного зв'язку.[9] Наприклад, рекурентна мережа може складатися з єдиного шару нейронів, кожен з яких спрямовує свій вихідний сигнал на входи всіх інших нейронів шару. Зображена на рисунку 2.11 рекурентна нейрона мережа та її розгорнутий вигляд.

Ідея RNN полягає у послідовному використанні інформації. У традиційних нейронних мережах входи і виходи незалежні. Але для багатьох завдань це не підходить. RNN називаються рекурентними, тому що вони виконують одну і ту ж задачу для кожного елемента послідовності, причому вихід залежить від попередніх обчислень. Ще можна сказати, що це мережі, які мають «пам'ять», яка враховує попередню інформацію.

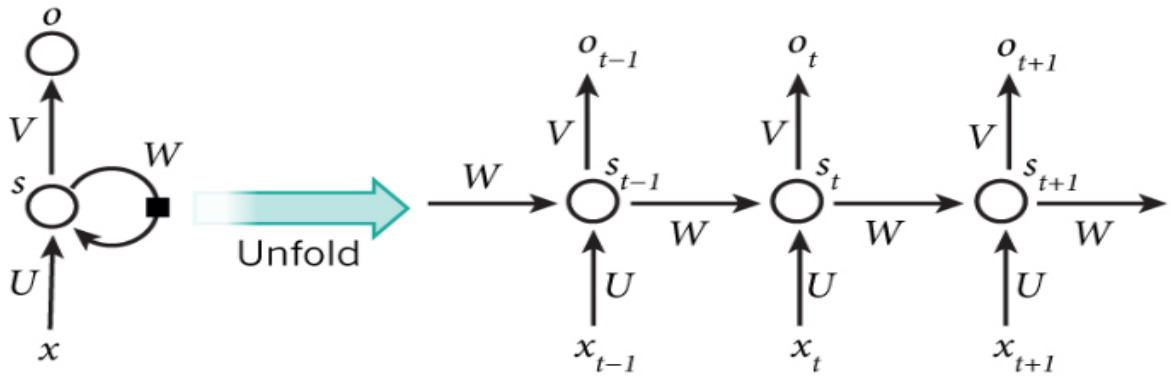


Рисунок 2.11 Рекурентна нейрона мережа та її розгортка

У рекурентних нейронних мережах кожен з нейронів у прихованих шарах отримує на вхід дані з певною затримкою в часі.[9] Також рекурентна нейронна мережа має стан, набутий при обробці попередніх елементів послідовності. RNN мають внутрішні цикли (петлі), тому рішення виноситься з урахуванням самих даних, і навіть поточного стану мережі.

2.6 Основні типи нейронних мереж

Нейронні мережі з прямим зв'язком досить старі — підхід бере свій початок із 50-х років. вона дотримується наступних правил[10]:

а) всі вузли повністю підключені

б) активація тече від вхідного шару до вихідного, без зворотних петель

в) є один прихований шар

Це штучна нейронна мережа, в якій з'єднання між вузлами не утворюють цикл. Найчастіше цей тип мереж навчається шляхом зворотного поширення помилки. Нейронні мережі з прямим зв'язком складаються зі статичних нейронів, тому сигнал на виході мережі з'являється в той же момент, коли подаються сигнали на вхід. Організація мережі може бути різною. Якщо не всі її складові є вихідними, кажуть, що мережа містить приховані нейрони.[11] У конкретних задачах нейрони зазвичай бувають згруповані у шари. На рисунку 2.12 показано типову схему нейронної мережі з прямим зв'язком з одним прихованим шаром.

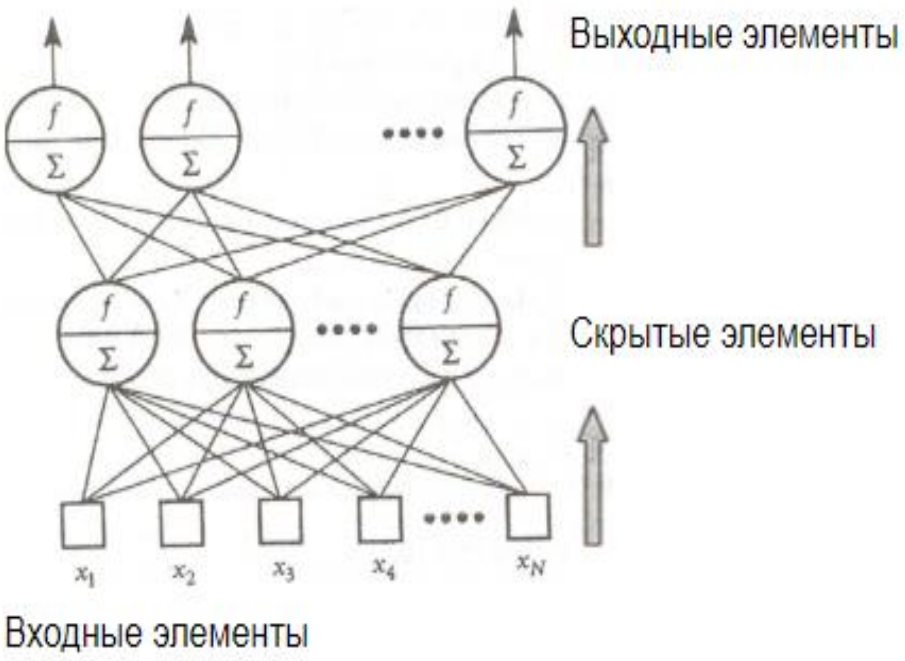


Рисунок 2.12 Типова схема нейронної мережі з прямим зв'язком з одним прихованим шаром.

Персептрон. Найпростіша і найстаріша модель нейрона, якою ми її знаємо. Приймає деякі вхідні дані, підсумовує їх, застосовує функцію активації та передає їх вихідному шару. Перцептрон складається з трьох типів елементів, а саме: сигнали, що надходять від датчиків, передаються асоціативним елементам, а потім елементам, що реагують.[12] Таким чином, перцептрони дозволяють створити набір «асоціацій» між вхідними стимулами та необхідною реакцією на виході. У біологічному плані це відповідає перетворенню, наприклад, зорової інформації на фізіологічну відповідь від рухових нейронів. Логічна схема перцептрону з трьома виходами зображений на рисунку 2.13.

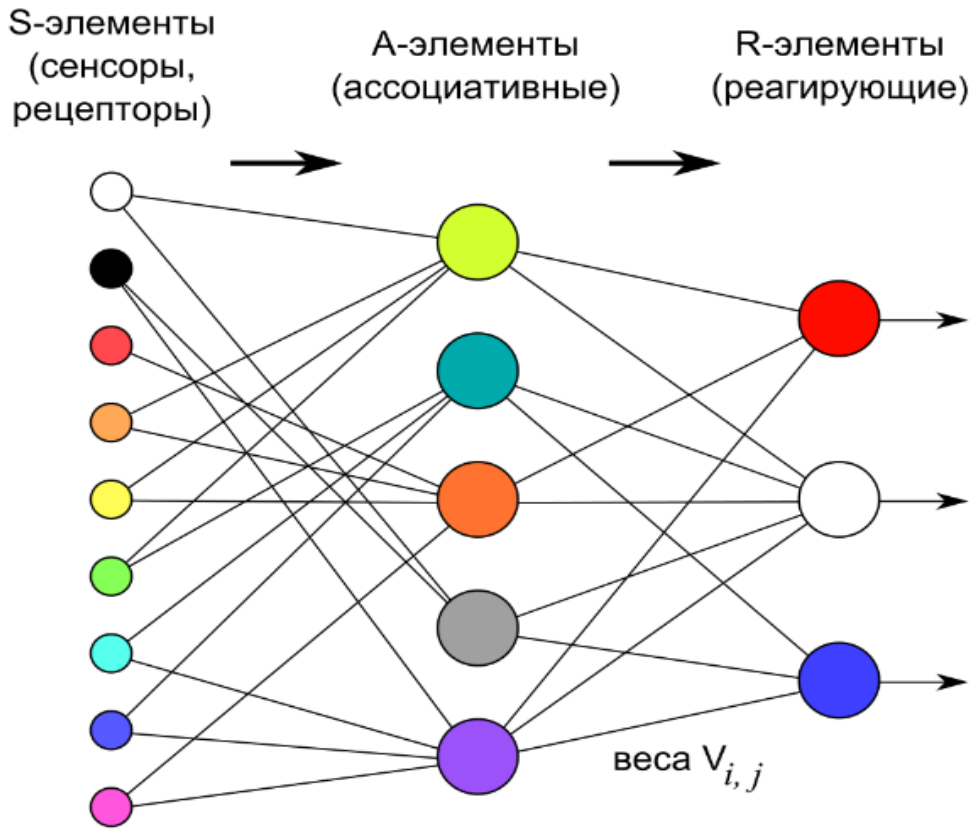


Рисунок 2.13 – Логічна схема перцептрону з трьома виходами

Елементарний перцептрон складається з елементів трьох типів: S-елементів, A-елементів та одного R-елемента. S-елементи – це шар сенсорів чи рецепторів. У фізичному втіленні вони відповідають, наприклад, світлочутливим клітинам сітківки

ока або фоторезисторам матриці камери. Кожен рецептор може бути в одному з двох станів — спокою або збудження, і тільки в останньому він передає одиничний сигнал у наступний шар, асоціативним елементам. А-елементи називаються асоціативними, тому що кожному такому елементу зазвичай відповідає цілий набір S-елементів. А-елемент активізується, як тільки кількість сигналів від S-елементів на його вході перевищила деяку величину. Сигнали від збуджених А-елементів, у свою чергу, передаються в суматор R, причому сигнал від і-го асоціативного елемента передається з вагою A-R зв'язку. Як і А-елементи, R-елемент підраховує суму значень вхідних сигналів, помножених на ваги. І якщо лінійна форма перевищує поріг, то видає «1», інакше «-1»

Довга короткострокова пам'ять (Long Short-Term Memory, LSTM) - незвичайна модифікація рекурентної нейронної мережі, яка на багатьох завданнях значно перевершує стандартну версію.[13] На рисунку 2.14 представлена архітектура мережі з довгою короткостроковою пам’ятью.

Майже всі вражаючі результати RNN досягнуто саме за допомогою LSTM.. При великому обсязі даних рекурентні нейронні мережі (RNN) стають просто непридатними, оскільки запам'ятовують швидше останню інформацію і забувають інформацію, отриманої давним-давно. Але бувають випадки, коли нам необхідно більше даних. Ця проблема схожа із загасаючими градієнтами в мережах прямого розповсюдження. Таким чином, розрив між актуальною інформацією та точкою її застосування може стати дуже великим. На жаль, у міру зростання цієї відстані, RNN втрачають здатність пов'язувати інформацію, тому на заміну звичайним рекурентним мережам приходить мережа з довгою короткостроковою пам'яттю.[14]

Нейронні мережі LSTM мають пам'ять тобто, поточна інформація зберігається для подальшого використання у майбутньому. LSTM є революційною технологією, яка використовується в багатьох програмах, наприклад, у віртуальному асистенті Siri від Apple. Застосування LSTM:

- розпізнавання мови
- оптичне розпізнавання символів
- генерація текстів

- прогнозування часових рядів
- машинний переклад
- питання-відповідь, чат-боти

LSTM розроблено спеціально, щоб уникнути проблеми довготривалої залежності. Запам'ятовування інформації на довгі періоди часу – це їхня звичайна поведінка, а не щось, чому вони намагаються навчитися.[13]

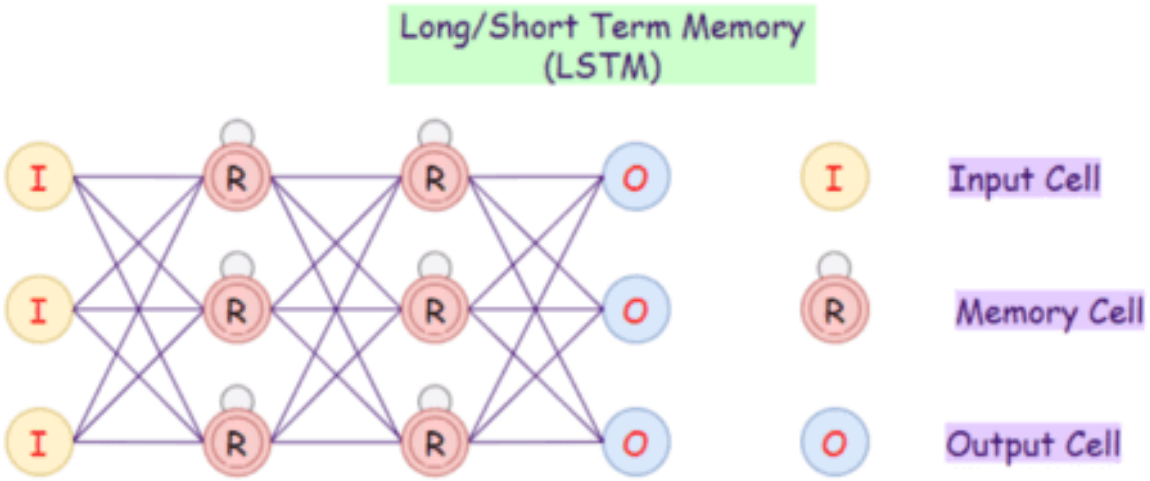


Рисунок 2.14 – Архітектура мережі з довгою короткостроковою пам’ятью

Мережа радіально-базисних функцій (Radial basis function, RBF) - це штучна нейронна мережа, яка використовує радіальні базові функції як функції активації.[15] Виходом мережі є лінійна комбінація радіальних базисних функцій входів та параметрів нейрона.[12] Мережа представлена на рисунку 2.15

Вона здатна класифікувати дані та робити передбачення. RBF-мережа приймає одне чи більше числових вхідних значень і генерує одне чи більше числових вихідних значень. Вихідні значення визначаються вхідними, а також набором параметрів, які називаються RBF-центроїдами, або центрами мас, набором RBF-інтервалів по ширині, набором вагових значень та набором зсувів.

При використанні RBF-мережі для класифікації та передбачення основна проблема – підібрати такий набір значень для центроїдів, інтервалів ширини,

вагових значень та зсувів, щоб вихідні значення, що були обчислені максимально відповідали набору відомих вихідних значень. Цей процес називається навчанням RBF-мережі.

Логістична функція відображає значення в діапазоні 0…1, відповідаючи на запитання так чи ні. Це добре для систем класифікації та прийняття рішень, але погано працює для безперервних значень. Навпаки, радіальні базисні функції відповідають питанням «наскільки далеко ми від мети?» [15]

Проблема цієї нейронної мережі – низька швидкість навчання. І навіть вона зберігає давню інформацію, тобто. не працює з урахуванням довгострокової перспективи. Звичайна рекурентна мережа, як і перцептрон, потрібна швидше для проектування складніших архітектур.

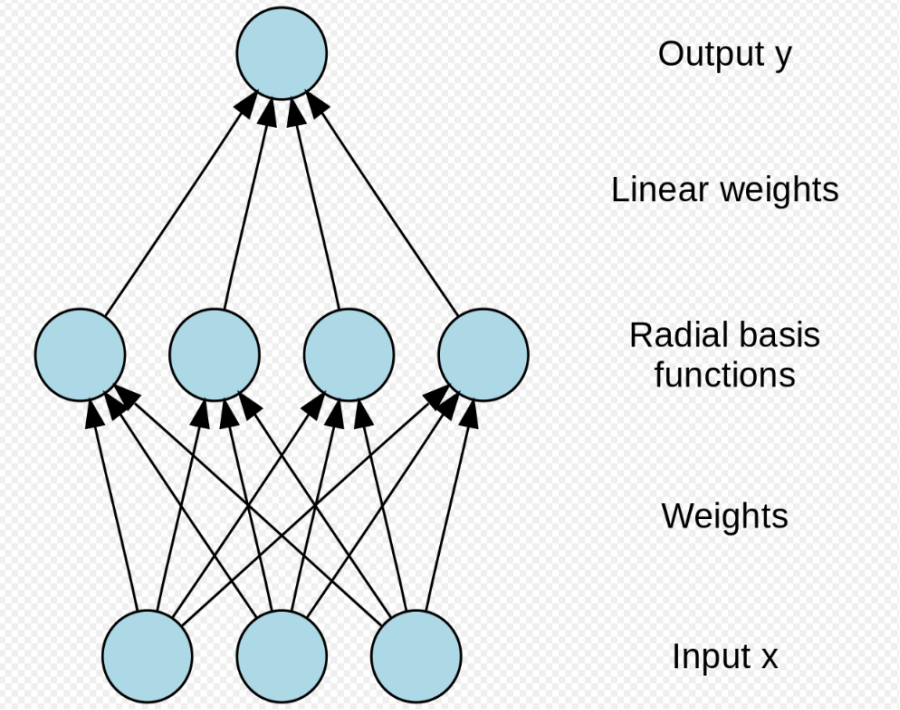


Рисунок 2.15 – Архітектура мережі радіальних базових функцій.

2.7 Навчання нейронної мережі

Прихований шар якраз і є тим шаром, в якому відбувається навчання. Вузли вхідного шару приймають вхідні сигнали, а вузли вихідного видають відповідь

мережі. Власне, прихований шар, або шари повинні навчити мережу перетворювати вхідні сигнали у відповідь.[16] Саме у ньому відбувається навчання. Насправді все навчання засноване на значеннях вагових коефіцієнтів до та після прихованого шару з метою зменшення різниці між бажаними та одержуваними векторами на виході.

Існує кілька основних типів навчання НМ:

- навчання з учителем – нейронна мережа навчається на розміченому наборі даних та передбачає відповіді, які використовуються для оцінки точності алгоритму на навчальних даних;
- навчання навчання з частковим залученням вчителя - є чимось середнім, воно використовує невелику кількість розмічених даних та великий набір нерозмічених;
- навчання без учителя - модель використовує нерозмічені дані, у тому числі алгоритм самостійно намагається отримати ознаки і залежність;
- навчання з підкріпленням - тренує алгоритм за допомогою системи заохочень. Агент отримує зворотний зв'язок у вигляді винагород за правильні дії;

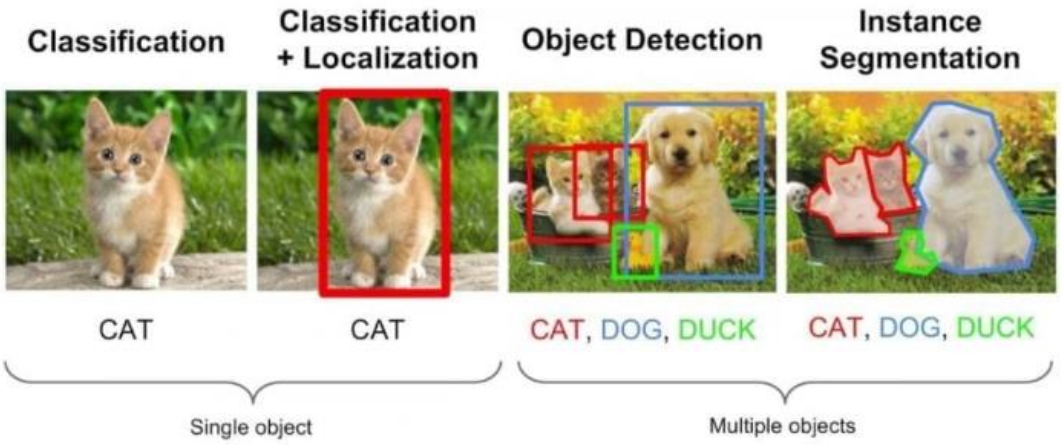
2.7.1 Навчання з учителем

Навчання з учителем (supervised learning) передбачає наявність повного набору розмічених даних для тренування моделі на всіх етапах її побудови.[16]

Наявність повністю розміченого датасета означає, що кожному прикладу в навчальному наборі відповідає відповідь, який алгоритм повинен отримати. Таким чином, розмічений датасет із фотографій кольорів навчить нейронну мережу, де зображені троянди, ромашки чи нарциси. Приклад зображено на рисунку 2.16. Коли мережа отримає нове фото, вона порівняє його з прикладами навчального датасета, щоб передбачити відповідь.

В основному навчання з учителем застосовується для вирішення двох типів завдань:

- класифікації;
- регресії.



а) - б)

Рисунок 2.16 – Приклад навчання з учителем а) класифікація б) подальше використання класифікації для сегментації та розпізнавання об’єктів

У задачах класифікації алгоритм передбачає дискретні значення, які відповідають номерам класів, до яких належать об'єкти. У датасеті з фотографіями тварин кожне зображення матиме відповідну мітку — «кішка», «собака» або «каченя». Якість алгоритму оцінюється тим, як точно він може правильно класифікувати нові фото з собакою і каченям.[16]

Завдання ж регресії пов'язані із безперервними даними. Один із прикладів, лінійна регресія, обчислює очікуване значення змінної y, враховуючи конкретні значення x. Більш утилітарні завдання машинного навчання задіють велику кількість змінних. Як приклад, нейронна мережа, що передбачає ціну квартири в Сан-Франциско на основі її площі, розташування та доступності громадського транспорту. Алгоритм виконує роботу експерта, який розраховує ціну квартири виходячи з тих самих даних.

Таким чином, навчання з учителем найбільше підходить для завдань, коли є значний набір достовірних даних для навчання алгоритму. Але так буває далеко не завжди. Проблема яка може бути – це нестача даних.

2.7.2 Навчання без учителя

Іноді перед алгоритмом стоїть завдання знайти наперед не відомі відповіді. У навчанні без вчителя (unsupervised learning) модель має набір даних, і немає явних вказівок, що з ним робити. Нейронна мережа намагається самостійно знайти кореляції даних, витягуючи корисні ознаки та аналізуючи їх. В залежності від задачі, модель систематизує данні за певними типами:

- кластеризація. Навіть без спеціальних знань людина може подивитися на колекцію фотографій та розділити їх на групи за видами птахів, спираючись на колір пера, розмір або форму дзьоба.[16] Саме в цьому полягає кластеризація – найпоширеніша задача для навчання без учителя. Алгоритм підбирає схожі дані, знаходячи загальні ознаки і групуючи їх разом. Приклад кластеризації наведено на рисунку 2.17
- виявлення аномалій. Банки можуть виявити шахрайські операції, виявляючи незвичайні дії у купівельній поведінці клієнтів. Наприклад, підозріло, якщо одна кредитна картка використовується в Каліфорнії та Данії в той самий день. Схожим чином, навчання без вчителя використовують для знаходження викидів даних.
- асоціації. Виберіть в онлайн-магазині підгузки, яблучне пюре та дитячий кухоль та сайт порекомендує вам додати нагрудник та радіоняню до замовлення. Це приклад асоціацій: деякі характеристики об'єкта корелюють з іншими ознаками. Розглядаючи кілька ключових ознак об'єкта, модель може передбачити інші, із якими існує зв'язок.

автоенкодери. Автоенкодери приймають вхідні дані, кодують їх, а потім намагаються відтворити початкові дані з отриманого коду. Не так багато реальних ситуацій, коли використовують простий автоенкодер. Але варто додати шари та можливості розширяться: використовуючи зашумлені та вихідні версії зображень для навчання, автоенкодери можуть видаляти шум із відеоданих, зображень або медичних сканів, щоб підвищити якість даних.

Рисунок 2.17 – Кластеризація даних на базі загальних ознак

У навчанні без вчителя складно обчислити точність алгоритму, оскільки «правильні відповіді» чи мітки відсутні. Але ці дані часто ненадійні або їх занадто дорого отримати. У таких випадках, даючи моделі свободу дій для пошуку залежностей, можна отримати кращі результати.

2.7.3 Навчання з частковим залученням вчителя

Навчання з частковим залученням вчителя (semi-supervised learning) характеризується тим, що навчальний датасет містить як розмічені, і нерозмічені дані. Цей метод особливо корисний, коли важко витягти з даних важливі ознаки або розмітити всі об'єкти – трудомістка задача.[16]

Цей метод машинного навчання є поширеним для аналізу медичних зображень, таких як скани комп'ютерної томографії або МРТ. Досвідчений рентгенолог може розмітити невелику підмножину сканів, на яких виявлено

пухлини та захворювання. Але вручну розмічати всі скани - надто трудомістке і дороге завдання. Тим не менш, нейронна мережа може отримати інформацію з невеликої частки розмічених даних і поліпшити точність передбачень у порівнянні з моделлю, що навчається виключно на нерозмічених даних.

Популярний метод навчання, який потребує невеликий набір розмічених даних, полягає у використанні генеративно-змагальної мережі (Generative adversarial network, GAN). Схема роботи GAN представлено на рисунку 2.18.

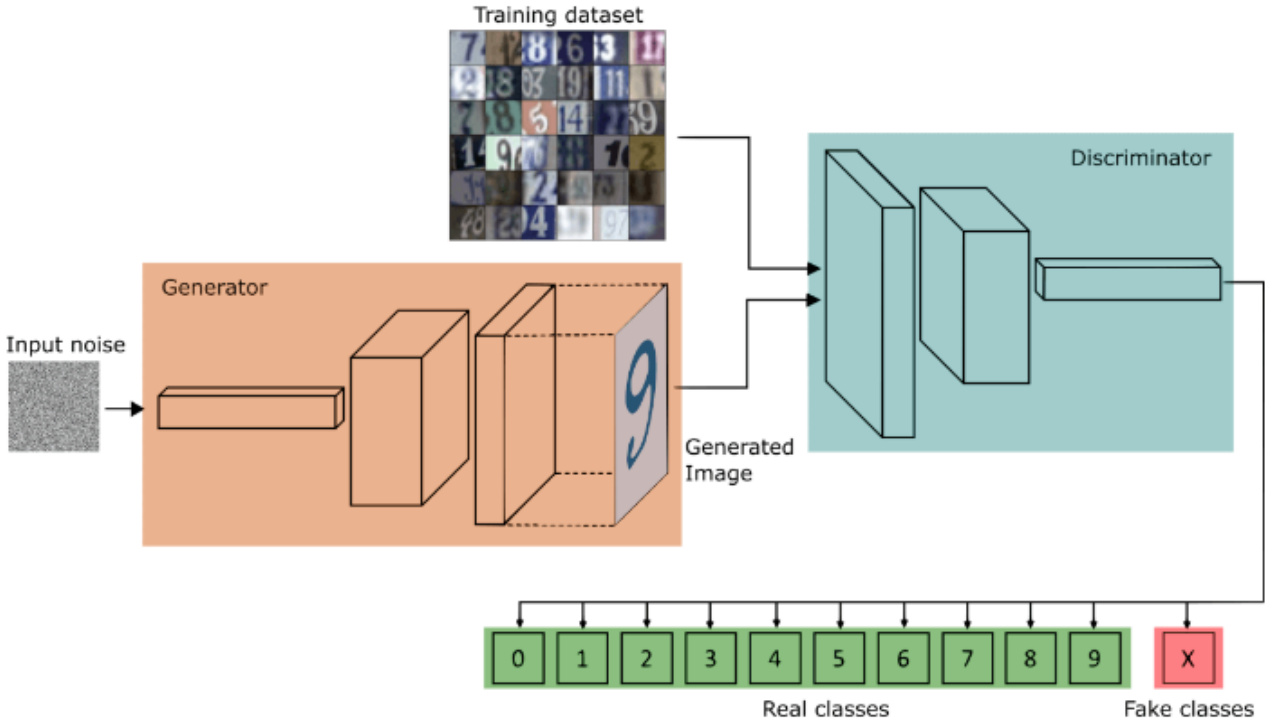


Рисунок 2.18 – Схема роботи GAN на прикладі виявлення зображеної цифри

Одна з мереж - генератор, намагається створити нові об'єкти даних, які імітують навчальну вибірку. Інша мережа – дискримінатор, оцінює, чи ці згенеровані дані є реальними чи підробленими.[16] Мережі взаємодіють і циклічно вдосконалюються, оскільки дискримінатор намагається краще відокремлювати підробки від оригіналів, а генератор намагається створювати переконливі підробки.

2.7.4 Навчання з підкріпленням

Відеоігри ґрунтуються на системі стимулів. Завершіть рівень та отримайте винагороду. Переможіть усіх монстрів та заробіть бонус. Потрапили у пастку – кінець гри, не влучайте. Ці стимули допомагають гравцям зрозуміти, як краще діяти у наступному раунді гри. Без зворотного зв'язку люди просто приймали б випадкові рішення і сподівалися перейти на наступний ігровий рівень.

Навчання з підкріпленням (reinforcement learning) діє за тим самим принципом.[16] Відеоігри – одне з найпопулярніших тестових середовищ для досліджень. Агенти ШІ намагаються знайти оптимальний спосіб досягнення мети чи покращення продуктивності для конкретного середовища. Коли агент робить дії, що сприяють досягненню мети, він отримує нагороду. Глобальна мета – передбачати такі кроки, щоб заробити максимальну нагороду зрештою. При ухваленні рішення агент вивчає зворотний зв'язок, нові тактики та рішення здатні призвести до більшого виграшу. Цей підхід використовує довгострокову стратегію — як і в шахах: наступний найкращий хід може допомогти виграти зрештою. Тож агент намагається максимізувати сумарну нагороду.

Це ітеративний процес. Чим більше рівнів із зворотного зв'язку, тим краще стає стратегія агента. Такий підхід є особливо корисним для навчання роботів, які керують автономними транспортними засобами або інвентарем на складі.[16]

2.8 Вибір нейронної мережі

Для нашої задачі, а саме створення action-гри з використанням алгоритму нейроеволюції наростаючих топологій для навчання штучного інтелекту, був вибраний персептрон. Тому що нам необхідна проста нейрона мережа, яка буде використовуватись алгоритмом нейроеволюції наростаючих топологій.

Перцептрон здатний навчатися і ґрунтується при цьому на статистичних даних. Інформація про образ розподіляється за ваговими коефіцієнтами, які в сукупності описують ті чи інші фрагменти образу.

Так як перцептрон ґрунтується на статистичному навчанні, то для нього доступні ті завдання, в яких об'єкти кожного класу мають спільні фрагменти, але можуть бути в різних комбінаціях.

Функцією активації обрали гіперболічний тангенс, має ті ж самі характеристики що і у сигмоїди, але вона не лінійна, а значить добре підходить для комбінації шарів, а також діапазон значень функції (-1,1). Активаційна функція не перевантажиться від великих значень, а у нас буде генерація нових поколінь.

В даній грі буде використане навчання з підкріпленням, адже воно найкраще підходить для ігор, а також допоможе швидше навчити наш штучний інтелект. За допомогою нагород та покарань штучний інтелект буде намагатись набрати якомога більше балів, тому не захоче помилятись.

3 НАВЧАННЯ ШІ ACTION-ГРІ З ВИКОРИСТАННЯМ АЛГОРИТМУ НЕЙРОЕВОЛЮЦІЇ НАРОСТАЮЧИХ ТОПОЛОГІЙ

3.1 Опис action-гри realUAcat

За допомогою середи розробки PyCharm була створена гра realUAcat. Ця гра жанру action, в якій головний герой котик, що стрибає. Його головна задача перестрибувати через перепони, в даному випадку квіти, що мають різну ширину. Котик по суті біжить на місті, а земля та квіти рухаються йому на зустріч, поступово гра пришвидшується. Якщо відбувається зіткнення з перепоною, то котик гине і гра починається наново.

Алгоритм нейроеволюції топологій, що наростають, розвиває нейронну мережу. На вході алгоритм приймає початкові значення вузлів, в процесі навчання випадковим чином додає нові вузли і утворює додаткові зв'язки між ними, в результаті чого формує на виході топологію нейронної мережі, що найбільш підходить для вирішення завдання. Особливістю NEAT є те, що він починає роботу зі спрощеною моделлю і ускладнює її лише у разі потреби.

Всього в одному поколінні за раз використовується 20 осіб. Рисунок 3.1 із 20 осіб лише один найуспішніший. В кожному поколінні коти розділяються на невеличкі групи – види, за подібністю розвитку та діяльності осіб. Наприклад деякі стрибають без перерви, а деякі просто біжать. В процесі забігу вивчаються дії котів, що приводять до найкращого результату. Коли всі коти натрапили на перешкоду, вони розташовуються у рейтингу за спаданням набраних балів. На даному етапі якраз і відбувається штучний «природний відбір».

Далі відбувається створення наступного покоління. Найкращі особи попереднього покоління стають батьками для наступного з невеличкими мутаціями.

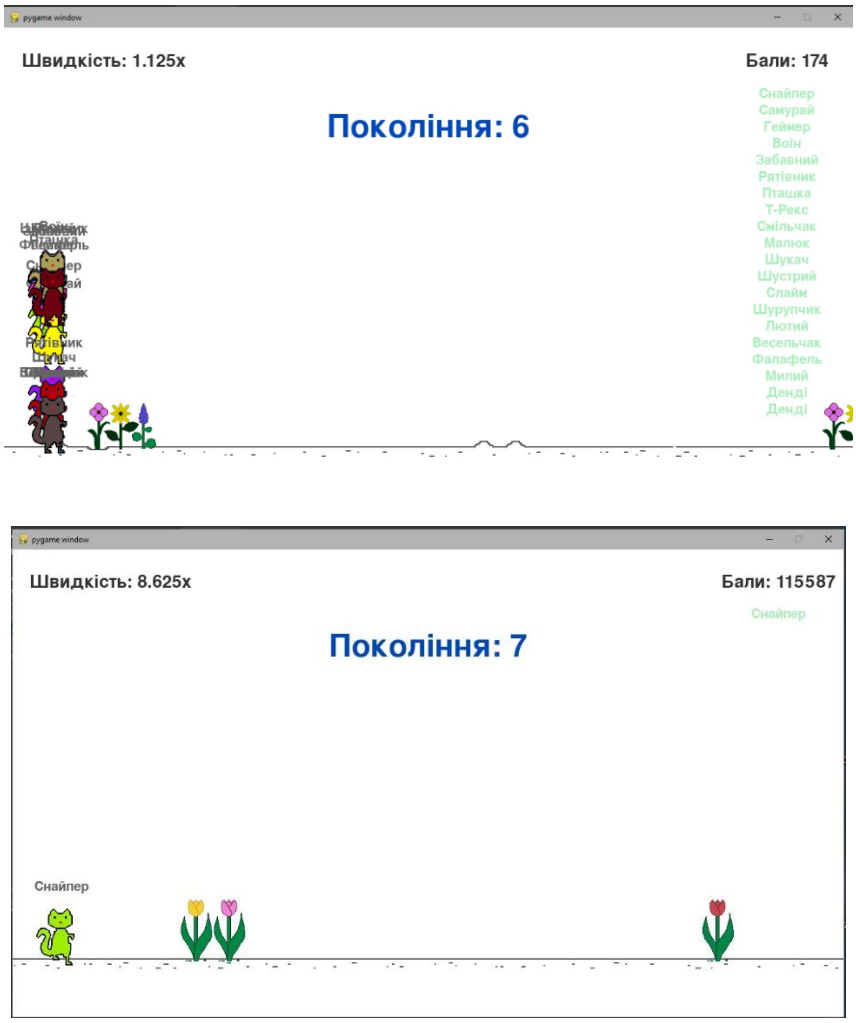


Рисунок 3.1 – Із 20 котиків на початку покоління лише один найуспішніший

Після першого запуску коти робили випадкові дії, одні лише стрибали, інші тільки бігли, треті робили і те і те, але не завжди вчасно і гинули. Все це можна побачити на рисунку 3.2 різноманітні дії котів при перших запусках. Через пару поколінь котики навчились перестрибувати квіти, що стали частіше з’являтись. І приблизно на 15 поколінні успішними були дії вже не одного кота, а чотирьох на великій швидкості, якій людина вже просто фізично не зможе зреагувати.

Натренувавши штучний інтелект на даній грі, його можна вбудувати як автопілот в автомобілі. Надлюдська реакція допоможе уникнути багатьох аварій, адже штучний інтелект з легкістю проаналізує ситуацію показавши неймовірну швидкість.

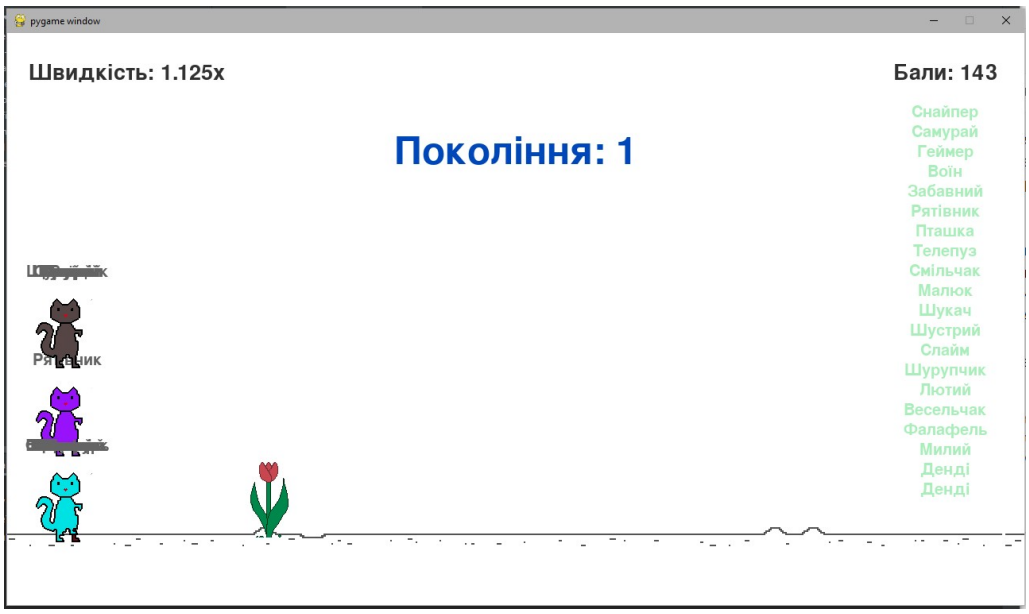


Рисунок 3.2 – Різноманітні дії котів при перших запусках

3.2 Модель нейронної мережі

Модель навчалась на основі перцептрону. Рисунок 3.3 Схема НМ нашої гри. На вхідному шарі маємо такі чотири вузли:

- позиція аватару;
- швидкість гри;
- дистанція до перепони;
- ширина перепони.

На виході один – стрибок. Для оцінки отриманого на виході числа необхідна функція активації, в ролі якої була використана функції гіперболічного тангенса, вона приводила всі вихідні значення в проміжок від -1 до 1. Навчання з підкріпленням є найкращим варіантом для нашої гри. Так як фітнес-функція визначає придатність особи, вона дорівнює пройденому шляху і додає 1 бал за кожну секунду процесу гри. За кожен стрибок без причини -1 бал, а за кожну позитивно пройдену квітку +5, а зіткнення з квіткою -10 балів. Таким чином нейронна мережа має навчитись значно швидше.

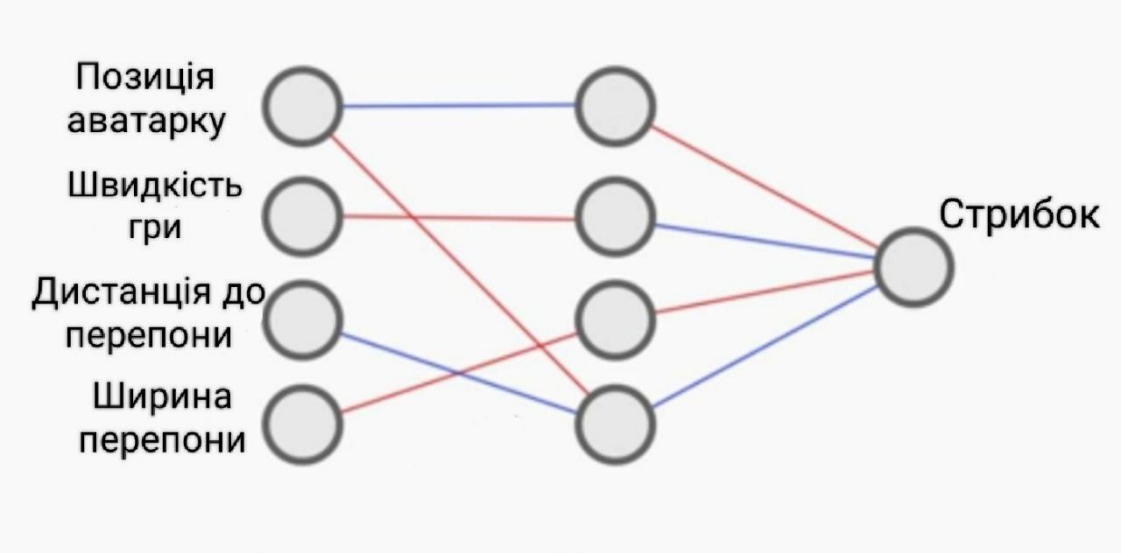


Рисунок 3.3 – Схема НМ нашої action-гри

3.3 Алгоритм NEAT

Алгоритм нейроеволюції наростаючих топологій, що є особливо ефективним внаслідок 3 ключових підходів:[17]

- усвідомлене схрещування генів внаслідок використання історичного походження.

- захист від зникнення значних генів у процесі відбору в результаті поділу генів на окремі види.

- старт з мінімальної кількості зв'язків та збільшення їх числа лише у разі потреби.

Алгоритм NEAT створено спеціально для реалізації вищезазначених принципів. NEAT (NeuroEvolution of Augmenting Topology) – це генетичний алгоритм, який може змінювати параметри та морфологію нейронних мереж.[17] Це еволюційний алгоритм для створення штучних нейронних мереж.

NEAT-Python зберігаються окремі популяції геному. Кожен геном включає два набори генів, що описують, як побудувати штучну нейронну мережу:

1) Вузлові гени, кожен із яких представляє окремий нейрон;
2) Гени зв'язку, кожен із яких визначає єдиний зв'язок між нейронами.

Щоб розвинути метод вирішення проблем, користувач повинен надати функцію пристосованості для обчислення реального значення, що вказує на якість геному людини: ті, у кого кращі здібності, мають вищі бали. Алгоритм еволюціонує залежно від кількості еволюцій, заданих користувачем, і кожне покоління осіб є результатом відтворення та мутації найкращих особин попереднього покоління[17].

Операції репродукції та мутації можуть додавати в геном вузли та/або зв'язки, тому геном, отриманий за допомогою алгоритму, розширяються в процесі розвитку, а нейронна мережа стає складнішою. Коли досягається задана кількість ітерацій чи коли хоча б одна особа перевищує поріг придатності, вказаний користувачем, алгоритм завершується. Приблизна схема роботи генетичних алгоритмів зображено на рисунку 3.4

Рисунок 3.4 – Приблизна схема роботи генетичних алгоритмів

Труднощі полягають в реалізації кросовера, точніше як реалізувати кросовер двох мереж із різними структурами. NEAT використовує ідентифікаційний номер, нові і більш високі порядкові номери генеруються на кожному додатковому вузлі, для відстеження походження вузла. Коли в ході мутації з'являється новий ген, йому

надається збільшений на одиницю ідентифікаційний номер, що таким чином відображає хронологію виникнення кожного гена в системі. Ті гени, які походять від спільного предка, зіставляються для схрещування, і відбувається з'єднання вузлів із загальним походженням.

Небезпека мутації полягає в можливості зруйнувати перспективні в майбутньому гени, тому ґрунтуючись на історичному походженні геномів, поділяє їх на окремі види. Кількість зайвих і непересічних генів є мірою сумісності. Чим більше геноми не перетинаються, тим менше еволюційної історії вони ділять, тим менше вони сумісні. Якщо сумісність генів не досягає певної межі, вони відносяться до різних видів і не конкурують між собою. Найуспішніші особини кожного виду в майбутньому схрещуються, що в результаті призводить до заміни всіх популяцій та отримання потрібного результату на вихідному шарі.

Інша потенційна складність – структурні варіації. У порівнянні з варіаціями ваг з'єднань, додавання вузла або з'єднання є перспективним у майбутньому, але в короткостроковій перспективі це може мати руйнівні наслідки, поки не буде виправлено менш руйнівною зміною[17]. NEAT ділить геноми на кілька видів для вирішення вищезазначених проблем. Чим вище схожість між генами, тим ближче геномна відстань, і група схожих генів є одним видом, і між видами існує жорстка конкуренція.

3.4 Опис файлу конфігурації алгоритму NEAT

Тут маємо декілька розділів. У розділ [NEAT] вказуються параметри, що стосуються загального алгоритму NEAT або самого експерименту. Цей розділ завжди є обов’язковим.[17]

fitness_criterion - функція, яка використовується для обчислення критерію завершення на основі набору придатності геному. Допустимі значення min, max, mean.

fitness_threshold - коли придатність обчислена, fitness_criterion досягає або перевищує цей поріг, процес еволюції припиняється із викликом found_solution методу будь-якого зареєстрованого класу звітності.

pop_size – кількість особин у кожному поколінні.

reset_on_extention – приймає значення True або False. Якщо True, то коли всі види одночасно вимирають через стагнацію, то буде створено нову випадкову популяцію. Рисунок 3.5 показує які дані ми внесли в конфігурацію NEAT.

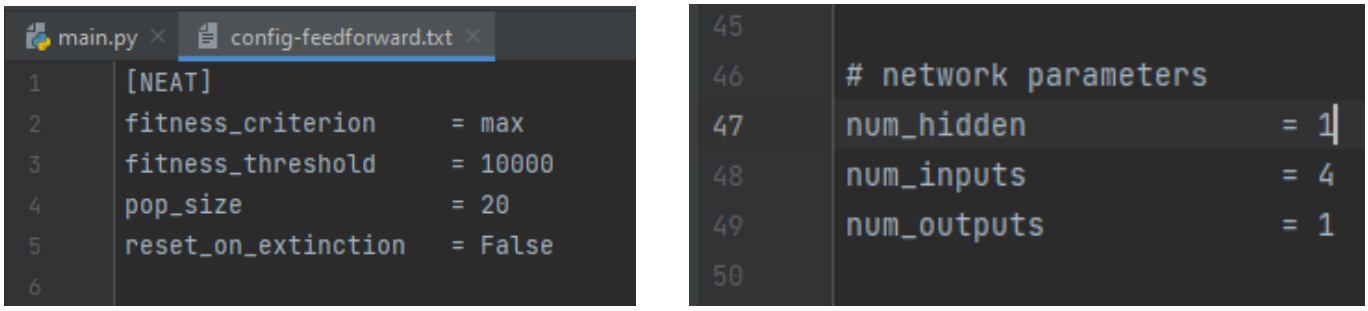


Рисунок 3.5 – Внесення даних в конфігурацію NEAT

Розділ [DefaultReproduction] – Цей розділ необхідний, лише якщо ви вказали цей клас як реалізацію відтворення під час створення екземпляра Config; інакше вам потрібно включити будь-яку конфігурацію (якщо така є), необхідну для вашої конкретної реалізації.

elitism - кількість найбільш придатних особин кожного виду, які збережуться такими, як є, від одного покоління до наступного. За умовчанням це значення 0.

survival_threshold - частка для кожного виду дозволила відтворити кожне покоління. За умовчанням це значення 0,2.

min_species_size – мінімальна кількість геномів на вид після розмноження. За умовчанням це значення 2.

Також є розділ [DefaultSpeciesSet] лише з однією функцією compatibility_threshold - особи, чия геномна відстань менша за цей поріг, вважаються належать до одного виду.[17]

Розділ [DefaultGenome] має багато функцій, опишемо деякі з них

activation_default - атрибут функції активації за замовчуванням, призначений новим вузлам. Якщо жодного не вказано або вказано «random», один із activation_options буде вибрано випадковим чином.

num_hidden – кількість прихованих вузлів в початковій популяції.

num_inputs – кількість вхідних вузлів, через які мережа отримує вхідні дані.

num-outputs – кількість вихідних вузлів, через які мережа доставляє виходи.

3.5 Середовище розробки PyCharm

PyCharm - це кроссплатформенне інтегроване середовище розробки мови програмування Python, розроблена компанією JetBrains на основі IntelliJ IDEA. Надає користувачеві комплекс засобів для графічного налагоджувача та роботи з кодом. Продукт доступний у двох версіях: PyCharm Community Edition – безкоштовна версія, знаходиться під ліцензією Apache License, і PyCharm Professional Edition – розширена версія продукту, що має додаткову функціональність[18].

PyCharm був випущений на ринок інтегрованих середовищ розробки для створення конкуренції з PyDev проте, на даний момент PyCharm використовує PyDev для налагодження коду та більш поширеного середовища розробки Komodo IDE. Бета-версію було випущено в липні 2010 року, версію 1.0 було випущено трьома місяцями пізніше. Версія 2.0 вийшла 13 грудня 2011 року. Версія 3.0 була випущена 24 вересня 2013 року. PyCharm Community Edition, безкоштовну версію з відкритим вихідним кодом, було опубліковано 22 жовтня 2013 року[18]. У березні 2016 року JetBrains перейшла на підписну модель ліцензування, а разом із цим змінилася і нумерація версій. Тепер номер версії виглядає як YYYY.R, де YYYY — рік випуску, а R — випуск цього року.

Користувачі можуть писати свої плагіни, тим самим розширювати можливості PyCharm. Деякі плагіни з інших JetBrains IDE можуть працювати з PyCharm. Існує більше тисячі плагінів, сумісних з PyCharm.

IDE – це не просто редактор для коду. Він має набагато більше можливостей, що полегшують життя розробнику. В PyCharm є більшість функцій, важливих для програміста[18]. Ось лише деякі приклади, що в ньому можна робити:

- створювати проекти. Проект мовою програмування - це не просто створення файлу. Коли розробник створює проект у PyCharm, середовище виділяє під нього окрему папку, де зберігає все, що пов'язане з цим проектом[18]. Так потрібні файли та компоненти знаходяться під рукою. Структуру проекту PyCharm показує в лівій частині інтерфейсу і дає можливість у будь-який момент переключитися на файл, що цікавить всередині нього.

- писати код для Python. Всередині проекту можна створити файл у потрібному розширенні та писати в ньому код. Синтаксис підсвічується автоматично, причому параметри підсвічування можна налаштувати. До того ж PyCharm дає можливість одразу перевіряти правильність написання та виділяти помилкові моменти. Середовище допомагає писати чистіший код.

- запускати. IDE підключено до інтерпретатора або компілятора потрібної мови. Python — мова, що інтерпретується, і PyCharm може запустити її інтерпретатор. Тому код можна виконати прямо всередині середовища, для цього не знадобиться відкривати консоль або будь-яку сторонню програму. В інтерфейсі IDE є кнопка запуску: достатньо натиснути на неї і код запуститься[18]. Результат виконання програма покаже одразу – виведе у спеціальну консоль усередині середовища чи відкриє нове вікно.

- налагоджувати. У середовищі є інструменти налагодження коду. Наприклад, можна налаштувати режим налагодження так, щоб показувати значення різних змінних у будь-який час. Або зупинити виконання на конкретному рядку і дивитися, чи код працює в цьому моменті, — це допомагає знайти місце, в якому відбувається помилка. Є й покрокове виконання: програма виконує один рядок коду і зупиняється, щоб розробник міг перевірити, чи вона працює на цій ділянці.

- тестувати. Автоматичне тестування – одна з найпоширеніших сфер застосування Python. І робити це у PyCharm зручно. У середовищі є інструменти автоматичної генерації коду, і до неї легко підключити модулі для тестування[18].

- коригувати. Щоб код був чистим і красивим і його було легко читати, потрібно дотримуватись правил «хорошого тону» для розробників. PyCharm може стежити виконання цих правил. Він також може автоматично розставляти переноси рядків та відступи та доповнювати написане: людина вводить лише частину команди, а PyCharm вже пропонує підказки для її закінчення.

- встановлювати бібліотеки та фреймворки. Деякі версії PyCharm «з коробки» підтримують низку популярних фреймворків для мови, інші дають можливість швидко їх завантажити та встановити. Одна з можливостей середовища – швидко знайти потрібний фреймворк у мережі, завантажити та підключити до проекту. Це зручно: для встановлення та розгортання оточення не доводиться користуватися безліччю додаткових інструментів.

- встановлювати доповнення та плагіни. Для самого середовища також є модулі, що розширюють її функціональність. Їх можна встановити зсередини IDE. Приклади таких модулів - перевірка читання коду, підказки за допомогою штучного інтелекту, розстановка дужок, що відсутні, і багато іншого. Деякі плагіни змінюють інтерфейс, якщо розробнику не подобається стандартний. Інші розширюють можливості самого середовища. Після встановлення доповненням можна скористатися як частиною IDE.

- писати іншими мовами. PyCharm призначена для Python, але в ній є підтримка інших мов[18]. Наприклад, Python часто використовується у веб-розробці, тому IDE також підтримує JavaScript для браузера та SQL для баз даних. Крім JavaScript, підтримуються TypeScript і CoffeeScript, популярні JS-фреймворки, а також мови HTML і CSS для верстки. У середовищі можна використовувати шаблонизатори – спеціальні інструменти, які допомагають створювати шаблони для веб-сторінок. Мови шаблонізаторів PyCharm теж розуміє.

3.6 Бібліотека pygame

Pygame – бібліотека модулів для мови Python, призначена для розробки мультимедійних програм з графічним інтерфейсом, наприклад ігор.[19]

Спочатку Pygame був написаний Пітом Шиннерсом. Починаючи приблизно з 2004-2005 року підтримується та розвивається спільнотою вільного програмного забезпечення.

Pygame – маленька бібліотека. яка включає зручні інструменти для малювання, роботи із зображеннями, відео, спрайтами, шрифтами та звуком, для обробки подій клавіатури та миші. Сам Python дозволяє писати короткий та ясний код. І хоча сам Pygame не використовується для комерційної розробки ігор, все ж може бути використаним досвідченими програмістами для швидкого створення прототипу гри, щоб подивитися, як працюватиме. Після цього програма переписується іншою мовою. Іншими словами, перевага Pygame у легкому навчанні та швидкій розробці. Але є один мінус, тут відсутня емуляція фізичних явищ.

Загалом ігровий цикл можна розбити на три категорії:

- обробка введення;
- оновлення гри;
- рендерінг.

В обробці введення йдеться про все, що відбувається поза грою – ті події, на які вона має реагувати. Наприклад натискання клавіш на клавіатурі, кліки мишею.

Оновлення гри це зміна всього, що має змінитись протягом одного кадру. Якщо персонаж у повітрі, гравітація має потягнути його вниз. Якщо два об'єкти зустрічаються на великій швидкості, вони мають вибухнути.[19]

Рендерінг – на цьому етапі все виводиться на екран: фони, персонажі, меню. Все, що гравець повинен бачити, з’являється на екрані у потрібному місці.

Ще один важливий аспект ігрового циклу – швидкість його роботи. Термін FPS, який розшифровується як Frames Per Second, тобто кадри в секунду.[19] Він вказує на те, скільки разів цикл має повторитися за одну секунду. Це важливо, щоб гра не була надто повільною чи швидкою. Важливо, щоб гра не працювала з різною швидкістю на різних ПК. Якщо персонажу потрібно 10 секунд на те, щоб перетнути екран, ці 10 секунд повинні бути постійними для всіх комп’ютерів.

Імпортуємо бібліотеку та задаємо базові настройки гри, такі як ширина та висота ігрового екрану та його колір (приклад 3.1)

```
import pygame

width = 1280 #ширина ігрового вікна
height = 720 #висота ігрового вікна
bg = (255, 255, 255, 255)#колір ігрового вікна
```

Приклад 3.1 Імпорт бібліотеки та задання базових настройок гри (файл main.py)

Далі створюємо гру та вікно. pygame.init() - це команда, яка запускає pygame. screen – вікно програми, яке створюється, коли ми задаємо його розмір у налаштуваннях. Далі необхідно створити clock, щоб переконатися, що гра працює із заданою частотою кадрів.[19] А також додаємо тут картинку дороги та шрифти для наших написів.(Приклад 3.2)

```
pygame.init()
screen = pygame.display.set_mode((width, height))
clock = pygame.time.Clock()
road_chunks = [
      [pygame.image.load('sprites/road.png'), [0, height - 100]],
      [pygame.image.load('sprites/road.png'), [2404, height - 100]]
]
font = pygame.font.SysFont("Roboto Condensend", 30)
score_font = pygame.font.SysFont("Roboto Condensend", 40)
name_font = pygame.font.SysFont("Roboto Condensend", 30)
heading_font = pygame.font.SysFont("Roboto Condensend", 70)
```

Приклад 3.2 Створюємо гру та вікно, додаємо картину дороги та шрифт

Тепер створюємо ігровий цикл використовуючи цикл while = True. Тут пишемо введення процесу, оновлення та візуалізація нашої гри.

Рендерінг починається з команди `screen.fill (bg)`. Але цього не достатньо. Дисплей комп'ютера працює не так. Змінити піксель – означає передати команду відеокарті, щоб вона передала відповідну команду екрану. За комп'ютерними мірками це дуже повільний процес. Якщо потрібно намалювати велику кількість об’єктів на екрані, це займе багато часу. Виправити це можна оригінальним способом, який називається подвійна буферизація.[19]

Подвійна буферизація, це ніби ви маєте двосторонню дошку, яку можна

повертати, показуючи то одну, то другу сторону. Одна буде дисплеєм – це те, що бачить гравець, а друга залишатиметься прихованою, її зможе «бачити» тільки комп'ютер. З кожним кадром рендеринг відбуватиметься на задній частині дошки. Коли малюнок завершується, дошка повертається та її вміст демонструється гравцю.[19]

А це означає, що процес відтворення відбувається один раз за кадр, а не при додаванні кожного елемента.

У pygame це відбувається автоматично. Потрібно лише сказати дошці, щоб вона перекинулася, коли малювання завершено. Ця команда називається flip(). Після того як намалювали всі необхідні об’єкти пишемо `pygame.display.flip()`. Головне, щоб функція flip() була в кінці. Якщо спробувати відобразити щось після повороту, цей вміст не з'явиться на екрані.

Також не забуваємо додати `pygema.QUIT` – подія, яка стартує після натискання хрестика та передає значення False, внаслідок чого ігровий цикл закінчується.[19]

І нарешті `clock.tick()`. Команда tick() просить pygame визначити, скільки займає цикл, а потім зробити паузу, щоб цикл, тобто цілий кадр, тривав потрібен час. У нас встановлено значення FPS 60, це означає, що довжина одного кадру — 1/60, тобто 0,06 секунди. Якщо цикл коду займає 0,04 секунд, тоді pygame зробить паузу на 0,05 секунд.

ВИСНОВКИ

Метою магістерської кваліфікаційної роботи є використання еволюціонуючої нейронної мережі в action-грі.

В роботі проводився аналіз архітектур нейронних мереж, їх типи, а також функції активації. На основі проведеного анаізу була обрана архітектура нейронної мережі, яка є первинною або простою, адже алгоритм нейроеволюціїї топологій буде її розвивати в процесі навчання, випадковим чином додасть нові вузли і утворить додаткові зв'язки між ними. Даною архітектурою став багатошаровий персептрон з одним прихованим шаром, та функцією активації гіперболічний тангенс.

Окрім цього було проаналізовано схожість роботи біологічного мозку з нейронною мережею. Також було розкрито що собою представляє нейронна еволюція та її алгоритми. Досліджено роботу алгоритму нейроеволюції наростаючих топологій – NEAT, який був використаний в роботі. Створена action-гра з використанням алгоритму нейроеволюції наростаючих топологій та прослідовно за скільки поколінь штучний інтелект навчиться в неї грати.

NeuroEvolution of Augmenting Topology (NEAT) є особливим, бо він на початку роботи працює зі спрощеною моделлю, яку надав користувач і ускладнює її лише тоді, коли це потрібно.

В даній роботі в одному поколінні приймають участь 20 осіб. Цей алгоритм в кожному поколінні розділяє особи на невеличкі групи за видами (за подібністю розвитку та діяльностю). В процесі одного покоління вивчаються найкращі результати а саме які дії привели до такого результату. Штучний «природний відбір» відбувається коли всі особи потрапили на перешкоду та вистроїлись в порядку зростання в залежності від набраних балів. Далі створення нового покоління та мутації.

Зіштовхнулись із проблемою стрибання осіб без потреби, але її вдалось вирішити за допомогою системи заохочення та покарання. До того ж, як наслідок, штучному інтелекту вдалось навчитись за меншу кількість поколінь.

Приблизно вже на 3-5 поколінні штучний інтелект з легкістю справлявся із поставленою задачею на високих швидкостях, де б людина вже не змогла реагувати.

У штучного інтелекту виробляється надлюдська реакція і ця характеристика може дуже допомогти людству. Після навчання такої нейронної мережі її можна вбудувати, наприклад, в автопілот для автомобіля. Штучний інтелект під час поїздки зможе сканувати всі перешкоди, що знаходяться навколо, так само, як і кіт дивився на квіточку, і вибирати найкращий напрямок руху автомобіля. Для застосування в Data science велику роль зіграє зміна ваги зв’язків і виявлення найважливіших ознак пошуку інформації. У разі котика зверталась велика увага на дистанцію до перепони та ширину перепони, але було всеодно на швидкість.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Action, 2022, URL: https://ru.wikipedia.org/wiki/Action
2. Шубко Н., Искуственный интелект, 2021, URL: https://teletype.in/@ndrezon/hvixiufKp
3. Job Talle Neuroevolution in squids 2020 URL: https://jobtalle.com/neuroevolution_in_squids.html
4. Нильсен М.А. Нейронні мережі та глибоке навчання, Determination Press, 2015.
5. Рашид Т., Создаем нейронную сеть, Санкт-Петербург,2017
6. Хайкин С., Нейронный сети. Полный курс. 2-е издание, Москва, 2006.
7. Милютин И., Функции активации нейросети: сигмоида, линейная, ступенчатая, ReLu, Than, 2018, URL: https://neurohive.io/ru/osnovy-data-science/activation-functions
8. Модель нейромереж, 2021, Модель URL: https://www.ibm.com/docs/ru/spss-modeler/saas?topic=networks-neural-model
9. Isakov S. Рекуррентные нейронные сети: типы, обучение, примеры и применение, 2018, URL: https://neurohive.io/ru/osnovy-data-science/rekurrentnye-nejronnye-seti/#pll_switcher
10. Романов В. П. Интеллектуальный информационные системы в эклномике, Глава 13. Нейронные сети с прямой связью ,URL: https://studfile.net/preview/986741/page:3/
11. Korablyov M., Axak, N., Soloviov D., Hybrid evolutionary decision-making modal based on neural network and immune approaches (2018 IEE 13th International Scientific and Technical Conference on Computer Sciences and Informayion Technologies, CSIT 2018
12. Andrew Tch, The mostly complete chart of Neural Networks, explained, 2017 URL: https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

13. Christopher Olah, Understanding LSTM Networks, 2015, URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

14. Кораблев Н.М., Фомичев А.А., Соловьев Д.Н., Чуприна А.А. Гибридные модели принятия решений с использованием иммунного похода, Информационные управляющие системы и технологии. Проблемы и решения: монография. Под науч. ред. проф. Вычужанина Владимира. – Одесса, 2019.

15. Джеймс Маккафри, Обучение сетей на основе радиально-базисных функций, 2015, URL: https://learn.microsoft.com/ru-ru/archive/msdn-magazine/2013/december/test-run-radial-basis-function-network-training

16. Беликова К. Обучение нейросети с учителем, без учителя, с подкреплением – в чем отличие? Какой алгоритм лучше?, 2018 URL: https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem

17. NEAT Overview URL: https://neat-python.readthedocs.io/en/latest/neat_overview.html

18. PyCharm, 2022, URL: https://blog.skillfactory.ru/glossary/pycharm/

19. Библиотека Pygame. Часть 1. Введение, 2021, URL: https://habr.com/ru/post/588605/.

20. Кораблев Н.М., Фомичев А.А., Соловьев Д.Н., Чуприна А.А. Гибридные модели принятия решений с использованием иммунного похода, Информационные управляющие системы и технологии. Проблемы и решения: монография. Под науч. ред. проф. Вычужанина Владимира. – Одесса, 2019.

21. Korablyov M., Axak, N., Soloviov D., Hybrid evolutionary decision-making modal based on neural network and immune approaches (2018 IEE 13th International Scientific and Technical Conference on Computer Sciences and Informayion Technologies, CSIT 2018.